JPL Publication 89-7, Vol. III

Proceedings of the NASA Conference on Space Telerobotics

Volume III

G. Rodriguez
H. Seraji
Editors

January 31, 1989

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 89-7, Vol. III

Proceedings of the NASA Conference on Space Telerobotics

Volume III

G. Rodriguez
H. Seraji
Editors

January 31, 1989

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT
NASA Conference on Space Telerobotics

These proceedings contain papers presented at the NASA Conference on Space Telerobotics held in Pasadena, January 31-February 2, 1989. The Conference was sponsored by the NASA Office of Aeronautics and Space Technology, together with ARC, LRC, GSFC, JSC, MSFC, KSC and JPL. The theme of the Conference was man-machine collaboration in space. The Conference provided a forum for researchers and engineers to exchange ideas on the research and development required for application of telerobotics technology to the space systems planned for the 1990s and beyond. The Conference: (i) provided a view of current NASA telerobotic research and development; (ii) stimulated technical exchange on man-machine systems, manipulator control, machine sensing, machine intelligence, concurrent computation, and system architectures; and (iii) identified important unsolved problems of current interest which can be dealt with by future research. There were about 500 international participants including about 100 from abroad.

An international program committee was established for the conference. A.K. Bejczy and H. Seraji of JPL acted as co-chairs for this committee. Members of the committee were

J. Amat, University of Barcelona, Spain
G.A. Bekey, University of Southern California
P.R. Belanger, McGill University, Canada
R.C. Bolles, Stanford Research Center
J.G. Bollinger, University of Wisconsin
W.J. Book, Georgia Institute of Technology
J.M. Brady, Oxford University, UK
F.E.C. Culick, California Institute of Technology
R.J.P. deFigueiredo, Rice University
W.R. Ferrell, University of Arizona
E. Freund, University of Dortmund, FRG
A.A. Goldenberg, University of Toronto, Canada
R. Jain, University of Michigan
T. Kanade, Carnegie-Mellon University
I. Kato, Waseda University, Japan
A.J. Koivo, Purdue University
P.D. Lawrence, University of British Columbia
J.Y.S. Luh, Clemson University
H.E. Rauch, Lockheed Palo Alto Research Lab
A. Rovetta, Polytechnic University of Milan
G.N. Saridis, Rensselaer Polytechnic Institute
T.B. Sheridan, Massachusetts Institute of Technology
L. Stark, University of California, Berkeley
D. Tesar, University of Texas at Austin
H. Van Brussel, Catholic University of Leuven
R.A. Volz, Texas Tech University

The Conference was organized by the Telerobotics Working Group of the NASA Office of Aeronautics and Space Technology. M. Montemerlo of NASA Headquarters and S.Z. Szirmay co-chair this working group. Representatives to this group from NASA centers and other research organizations are

D. Akin, Massachusetts Institute of Technology
J. Bull, Ames Research Center
R. Davis, Kennedy Space Center
S. Fisher, Ames Research Center
J. Haussler, Marshall Space Flight Center
A. Meintel, Langley Research Center
J. Pennington, Langley Research Center
D. Provost, Goddard Space Flight Center
C. Price, Johnson Space Center
L. Purves, Goddard Space Flight Center
C. Ruoff, Jet Propulsion Laboratory
E.C. Smith, Marshall Space Flight Center
M. Zweben, Ames Research Center

ACKNOWLEDGMENTS

Acknowledgments are due to R. Doshi, D. Diner, J. Barhen and A. Fijany of JPL and Professors L. Stark of UCB and H. Stephanou of George Mason University for their help in organizing invited technical sessions and discussion panels. Appreciation is due to P. McLane for setting up registration and conference facilities, and to L. Anderson for technical editing of the proceedings. Acknowledgments are also due Donna L. Milton of JPL for handling the local arrangements and coordination and administrative aspects of the Conference.

CONTENTS

Volume I

VOLUME III

VOLUME IV

VOLUME V

PLENARY SESSION

THE FLIGHT TELEROBOTIC SERVICER: NASA's FIRST OPERATIONAL SPACE ROBOT

Charles F. Fuechsel
FTS Project Manager

Goddard Space Flight Center
Code 409
Greenbelt, MD 20771

INTRODUCTION

The application of robotics to operations in space promises great benefits in extending the capability of man in exploration. Robotics is highly relevant to NASA's long-range vision of establishing a manned base on the moon, and later in exploring the surface of mars. An interplanetary voyage is of a different order of complexity from traveling to earth's moon because rapid return to earth in the event of trouble would be impossible. For such a mission involving humans, it would be prudent to first establish an operational base at the destination complete with an earth return capability. Humans would not begin their voyage until the remote base was assembled and known to be operational. Assembly and maintenance of such a base might be performed by robots that would work with a high degree of autonomy. Alternatives to the exploration of mars by direct human presence are under consideration by both the United States and the Soviet Union. In these concepts, autonomous surface vehicles would navigate the planet performing a variety of detailed exploratory functions such as mapping, seismic measurements, sample collection and analysis. Both of these approaches to the exploration of mars depend to a high degree on the ability of robotic machinery to perform complex functions without real-time human direction. Closer to home and in time, robotics will begin to play a role in space operations in the construction and maintenance of Space Station Freedom. This paper will introduce the Flight Telerobotic Servicer Project as an element of the Space Station Freedom, and discuss its objectives and some special challenges it faces.

THE NEED FOR ROBOTICS IN BUILDING THE SPACE STATION

The Space Station Freedom will be assembled on orbit through a sequence of 26 shuttle launches, each bringing a full load of truss members, modules, and work fixtures. The assembly process will be performed by astronauts wearing pressure suits - termed Extra-Vehicular Activity or EVA. The Shuttle's remote manipulator arm (RMS) will provide mobility for both crew and equipment. It is estimated that several hundred hours of astronaut EVA time will be required to complete assembly of the initial configuration of the space station.

Presently, EVA is not performed during the first two days of shuttle flights to allow the crew to acclimate to weightlessness and the tendency toward disorientation which EVA makes worse. After the two day adjustment period, each EVA session requires two hours of preparation time before the actual EVA operations begin, and the useful operation time is limited to about 7 hours due to the physical demands of flexing a fairly stiff pressure suit and exhaustion of consumables such as cooling water. New suit technology employing more compliant joints and closed cooling systems are currently being investigated to reduce these limitations. Each EVA session places the crew at some degree of risk from failure of one of the essential parts of the EVA system. These systems are designed to be highly reliable, but as the number of EVA hours worked increases, even small failure probabilities multiply and become a significant risk in the long term.

Considering all of these factors, it can be seen that robotic machines, designed to perform a significant portion of the space station assembly, would reduce the requirement for manned EVA, thereby providing significant gains in productivity and safety for the crew. This is one of the principle objectives of the FTS Project. The method of controlling the telerobot will initially be teleoperation where a human operator within a pressurized module of the space station continuously directs every step of the robot's operation and performs all object recognition and maneuver planning.

OTHER USES FOR A SPACE TELEROBOT

There are important applications for robotics beyond assembly and maintenance in proximity of the Space Station. When attached to a suitable transfer vehicle such as the Orbital Maneuvering Vehicle (OMV), the FTS will be capable of visiting spacecraft in orbits which the Shuttle, and hence astronauts, cannot presently achieve. This opens the new possibility of performing remote servicing operations on space assets without being limited by the shuttle operational envelope. However, because of the inherent communications delays and likely bandwidth limitations associated with such a mission, a different operational mode for the FTS will be required. Called supervised autonomy, this mode must provide the capability for the FTS to perform many operations with little or no human direction. The human supervisor would intervene only at planned stages of the operation to confirm the success of prior steps performed autonomously by the robot, and to confirm readiness to proceed to the next step. Autonomous operation of the telerobot would also be valuable in the space station assembly and maintenance application because it would relieve the human operator from continuously directing every action to be performed, thereby freeing the operator to do other tasks.

THE TECHNOLOGY CHALLENGE

Achieving autonomous control capability will require that most of the planning and sensor interpretation functions that are done by a human operator in the teleoperation case must now be performed by sensors and processing aboard the telerobot. A substantial on-board database is also needed to provide rapid access to geometric descriptions of the workspace, procedures to be executed, and criteria for success and failure for each step. The algorithms and machinery required to support autonomous operation of a robot are currently being explored in research laboratories, and much work is needed to develop them into operational capabilities. Therefore, autonomous operation is considered a long-term goal for the FTS, and initial capabilities will concentrate on teleoperation. This initial capability will include sensing and processing to support bilateral force reflection - a technique whereby forces and torques being exerted by the end effector are measured and used to actuate motors in the hand controller such that the operator "experiences" the forces and torques being exerted on the workpiece. This capability is needed to perform certain tasks where the ability to "feel" proper engagement is important to successful teleoperation. In addition, a few elementary forms of autonomous control will be provided, including active compliance and modes of control shared between the human operator via the work station and on-board algorithms.

But the attainment of significant degrees of autonomous control remains a long-term objective, and this implies that as robust sensing and processing techniques and hardware become available, the FTS must be upgraded to accommodate them. The FTS design anticipates this requirement, and employs an architecture that will facilitate the refinement or the replacement of function when required. The selected architecture is an emerging standard called NASREM developed jointly by NASA and the National Institute for Standards and Technology. [1]

THE OPERATIONAL CHALLENGE

There are two critical questions that need to be considered: The first is, "what can a telerobot do in space?" Our operational experience to-date for tasks involving a high degree of dexterity is that very little has been done robotically in space, and so a fundamental credibility must be established. The FTS Project has responded to this through a forum called the Mission Utilization Team whose charter is to analyze a representative suite of space station assembly tasks. A process of task decomposition develops the detailed sequence of steps required to perform a given task and identifies the needs for mobility, utilities at specific locations, special fixtures, and use of standard facilities such as the Remote Manipulator System. The analysis proceeds down to the geometric descriptions of the work space and the kinematics

of the FTS. Positioning of the robot and the objects it works with are worked out in time sequence and dynamic envelope clearance is checked at every incremental position through the entire task. The process is laborious and rigorous, and yields a good initial assessment of feasibility. However, full-scale mockup testing will be required to ultimately prove the ability of the system to accomplish the task.

The FTS Project includes two test flights aboard the Space Shuttle to evaluate the effectiveness of the human-machine interface and to demonstrate the ability of the telerobot to perform representative tasks. The experience gained through these flights will establish a significant degree of confidence in relying on telerobots to perform important space station assembly work.

The second question to be addressed is: "Believing that certain jobs can be performed by robots, what is the case for probable success for a specific task?" The FTS is to be an operational system that can be depended upon to perform important tasks in constructing and maintaining SS Freedom. Dependability and the required end result of accomplishment of function distinguish an operational system from an experiment in which the primary objective is information. Tasks to be performed by the FTS must have a substantial probability of working, even in the presence of difficulty. If it is not successful, then unplanned use of EVA as an alternative will come at potentially high expense in terms of the overall mission. The assurance of a high probability of success implies a significant investment in FTS reliability, diversity of ways of doing things, operations testing, and long-term technical support. Dependability in the context of operations is a quality that is not merely designed-in, but must also be earned through successful mission testing. The preparation for performing space operations using astronauts is presently a significant undertaking. Procedures and contingency plans must be developed, tested, and rehearsed thoroughly by the crew to assure that the mission will succeed even when things don't go as planned. The presence of a robot and its control station adds several new considerations to the mission planning process, including the attachment of the robot to the work site, planning & handling of reaction loads, and the interaction of the robot with other active control systems such as the shuttle or space station RMS. An effective space robotic system must include capabilities to support both mission planning and the conduct of simulations that interact with humans and other systems in a highly realistic manner.

For these purposes, the FTS Project includes the development of a trainer - a telerobot and workstation capable of operation in a 1-G environment, and which are highly representative of the FTS flight system. The project will develop mockups and simulators as required to accomplish pre-integration of the FTS with the

space station. A Robotics Development, Test, and Integration facility is also being prepared to support the testing of procedures and tooling to perform actual assembly and servicing operations and to evaluate advanced technology and control techniques. [2]

CONCLUSION

In being NASA's first general-purpose space telerobot, the FTS Project has simultaneously a significant technical challenge and a great responsibility to do the job well so that robotics becomes firmly established as an effective way to operate in space. We recognize that in being the first, a degree of skepticism must be overcome. And rightly so, because the NASA reputation for successful operations in space was not achieved by recklessly attempting untried techniques for the first time in orbit. Therefore a rigorous program is required to demonstrate the capabilities of the FTS together with it's control system and human operators to perform the actual tasks that it will be called upon to perform in orbit.

In order for FTS to achieve its long-term objectives in control autonomy, critical developments must be accomplished in the laboratory and then transferred to the FTS operational system. The research efforts sponsored by NASA should target these requirements, and the FTS Project must provide means to develop the results of these efforts into robust operational techniques and hardware. The FTS is being designed in a way to make such changes feasible, and the necessary development and integration facilities are planned to support this process. In the long term, FTS is only a beginning in applying robotic technologies to performing general physical tasks in space. It is inevitable that many robotic systems will follow, some similar to FTS in form and function, and others that are intended for quite different applications. It is our hope and commitment that the FTS will be the first positive step for robotics as an effective operational tool in spaceflight.

REFERENCES

[1] "The FTS: From Functional Architecture to Computer Architecture" R. Lumia, National Institute of Standards and Technology.

[2] "Development of a Robotics Technology Test Bed for the Flight Telerobotic Servicer Project" G. Mosier, M. O'Brien, and R. Schnurr, Goddard Space Flight Center.

FLEXIBLE ARMS

Modeling, Design, and Control of Flexible Manipulator Arms: Status and Trends

Wayne J. Book
The George W. Woodruff School
of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

The desire for higher performance manipulators has lead to dynamic behavior in which the flexibility is an essential aspect. This paper first examines the mathematical representations commonly used in modeling flexible arms and arms with flexible drives. Then design considerations directly arising from the flexible nature of the arm are discussed. Finally, controls of joints for general and tip motion are discussed.

1. MODELING FLEXIBLE ARMS

Models are used for simulation, analysis, and synthesis. In robotics, models may be used directly in the control algorithm with the computed torque technique. We will first look at the representation of the flexible behavior. Then, the incorporation of this flexible behavior into an overall arm model will be considered.

Modeling the Flexible Behavior

Examination of the energy storage characteristics of a component is helpful in assessing the modeling requirements of a system. Rigid arms store kinetic energy by virtue of their moving inertia and store potential energy by virtue of their position in the gravitational field. The flexible arm also stores potential energy by virtue of the deflection of its links, joints, or drives. Joints have concentrated compliance which is well modeled as a pure spring storing only potential energy. Drive components such as shafts or belts may appear distributed but store little kinetic energy due to their low inertia, and a lumped parameter spring model succeeds well for them also. Links are subject to torsion, bending, and compression. Torsion of a link stores potential energy but little kinetic energy due to low mass moment of inertia about the longitudinal axis of the beam and is thus well represented as a massless spring. Compression stores little potential energy due to the high compressional stiffness and dynamics along this axis is well described by a rigid mass. Links subject to bending store potential energy by virtue of their deflection as well as kinetic energy by virtue of their deflection rates and a good model must include this distributed nature. Partial differential equations result from an analysis of this type of problem, with time and one independent spatial variable usually adequate to represent the dynamic solution of the generally slender

links. The dynamics of the link itself may be represented by the Bernoulli-Euler beam equation:

$$EI \frac{\partial^4 y}{\partial x^4} + \mu \frac{\partial^2 y}{\partial t^2} = f(x,t) \qquad (1)$$

which ignores shearing of the beam and the mass moment of inertia of a differential element along the length. The Timoshenko beam equation includes these two effects and should be used if the beam is short relative to its diameter. Such "stubby" links are likely to be essentially rigid anyway. The Bernoulli-Euler beam may have a varying cross section, but analytical solutions are not available except for the simplest variations. Thus the partial differential equation is useful to accurately model only very simple real links, or to study the general nature of the problem. This is not a great limitation since most time domain analysis is performed on a finite dimensional approximation of the distributed parameter system anyway.

Other assumptions are usually made when the partial differential equation is employed. While body forces may be included on the right side of (1), these forces due to translation and rotation of the link are quite complex and are only represented in simple cases not representative of telerobotics, such as a constant spinning satellite with an antenna. As discussed below, PDE's with simple boundary conditions are also used to obtain a set of basis functions. These spatially discretized equations may include body forces in at least an approximate way.

If we treat time as a continuous variable, an ordinary differential equation can be obtained by representing the beam shape as an infinite sum over a set of basis functions, each multiplied by its own time varying amplitude. Suitable approximations result from discarding all but a finite number of these functions as in (2). For arms with only rotational joints this procedure is quite natural.

$$y(x,t) = \sum_{i=1}^{m} a(t)\, \Phi(t) \qquad (2)$$

Prismatic joints on flexible links are another problem. As a flexible link moves in and out of a fixed prismatic joint its length effectively changes. Physically viewed, the energy stored in the link deflection must be transferred to another part of the beam. This is easily illustrated by a vibrating hack saw blade retracted over the edge of a table. The peak strain in the blade during the vibration must increase to store the same energy in a shorter blade and the frequency of the blade increases due to its higher natural frequency. Real manipulators will have real joints which are not ideally constrained, and must be examined for the degree to which they are able to constrain the translation, rotation and curvature of the link. These joints will dissipate vibrational energy as well, in fact this may be their most desirable feature. Research on modeling prismatic jointed flexible manipulators is limited.[1] Variation of the mode shapes is one approach to the problem, but it will not be discussed in detail in the remainder of the paper.

In spite of the long standing use of a truncated series of shapes to represent the kinematics of a flexible beam, no unimpeachable rule has evolved on the selection of these shapes to obtain the needed combination of model simplicity and accuracy. Popular choices are

1. Simple polynomials
2. Eigenfunction solutions to simple eigenvalue problems
3. Eigenvector solutions to finite element problems
4. Modal test results on the actual components.

Qualitatively speaking, the best results seem to come from shapes which allow the natural shape of the link when in the total system to be accurately described. Thus an "augmented body" which has the mass and/or inertia of the other links represented as a rigid appendage on the end of a given flexible link can be used to get the shape functions for that given link. This has given rise to endless tinkering with the link boundary conditions in an attempt to find the "perfect approximation." Some of these are described by Craig[2]. For manipulator arms this incorporates a wide range of effects varying with

1. Payload
2. Contact with the environment
3. Joint position
4. Accelerations due to joint motion
5. Feedback control law and gains.

With so much uncertainty we must accept either a) models of high order, b) models of low accuracy, or c) models not appropriate for the full range of operation. In particular, decisions made on the basis of these models should recognize their approximate and specialized nature.

The choice of mode shapes modifies the "rigid" motion variables. Different choices have different advantages. A clamped boundary condition leads to a physically measureable joint variable and simpler coefficients of the joint torques.[3] Pinned-pinned boundary conditions lead to ease in specifying the location of the joint tip and have been used to advantage in computed torque control since the tip position is specified by one variable per link.[4] Others have used free-free boundary conditions and described the link c.g. with "rigid" motion variables.[5]

Large Motion Equations

Implicit in the above description of the arm flexible components is the assumption of small excursions from a nominal position. This assumption could be violated in several ways. Nonlinear material behavior would result from strains beyond the elastic limit, for example. Even with linear elastic material behavior, one end of a sufficiently long beam can rotate through multiple revolutions with respect to the other end in violation of the small

motion assumptions of the derivation. A straight beam of length L projects a distance L onto an axis x. When one end of the beam is deflected from that axis and the other end is kept tangent to the x axis, the projected distance is no longer L, but must be less. The length of the beam remains constant at L, assuming no axial extension. The shorter projection on x is ignored as a standard procedure, but has been shown by Ryan and Kane[6] to lead to obvious errors under reasonable conditions of rotation. Likewise, centrifugal stiffening, often ignored, creates terms that affect the system dynamics. While the conditions of rotation of their example is unlikely for space robots, the point at which these phenomena become important is not clearly known and a point for future research.

Given the distributed behavior of a flexible link and the lumped behavior of joints and drives, overall equations of motion can be derived by several methods. A finite dimensional model will be assumed here. Lagrange's equations and several variations thereon are readily applied. Kane has formulated equations in a manner of some distinction.[7] Newton's laws are less popular because of the distributed nature of the flexible links. Only the first, Lagrange's equations, are discussed at all here. Lagrange's equation for flexible links differ from rigid links because the flexible degrees of freedom appear principally as a sum rather than as a product due to the parallel contribution of each of the assumed shape functions. This is easily illustrated with the homogeneous 4x4 matrix formulation of the forward kinematics. For a rigid arm, the end point coordinate transformation is:

$$T = A_1A_2A_3A_4A_5A_6 \qquad (3)$$

where A_i is the transformation representing the joint and link. For the flexible arm the end point coordinate is

$$T = A'_1E_1A'_2E_2A'_3E_3A'_4E_4A'_5E_5A'_6E_6 \qquad (4)$$

$$E_i = H_i \sum_{j=1}^{m_i} \delta_{ij} M_{ij}$$

where H_i transforms along the length of the undeformed i-th beam and M_{ij} adds the effect of mode j. A'_i transforms for the joint only.

where the E_i's incorporate the summation of the assumed "mode" shape transformations $E_{i,j}$ and L_i incorporates the undeformed transformation of the link length. When an integration over the length of the flexible beam is used to incorporate all the kinetic and potential energy contributions, the assumed shapes are integrated to yield, for example, "modal masses", "modal stiffnesses", and "modal input matrix elements". These become coefficients in the final dynamic equations and are one way the choice of shape influences the final result. The choice of shape also will affect the calculation of outputs from the model, such as end point position or strain in the links. If a finite element or experimentally generated shape is used, equivalent integrals are approximated from the nodal points in the model.

The coefficients described above involve cross products of the various shape functions. The choice of orthogonal shape functions eliminates a number of terms from the final equations of motion, since for orthogonal functions

$$\int_0^{\ell} \Phi_i(x)\, \Phi_j(x)\, dx = 0, \quad i \neq j \qquad (5)$$

The orthogonality condition is automatically obeyed by the eigenfunctions of components. If boundary conditions are chosen to represent, for example, the mass of an augmented body the orthogonality condition will incorporate terms outside the integral. The true net value of this simplification is not clear. Operations such as inversion of the inertia matrix and multiplication by the inverse eliminate the zero coefficients. In simulation the coefficients may be recalculated relatively infrequently. Some researchers support the use of polynomials because they are able to represent more general conditions on the flexible link. Even though a polynomial is not orthogonal it is simple to compute. Incorporating the orthogonality condition correctly complicates the derivation procedure with the hope of ultimately reducing the complexity of the final equations. Since the equations are so complex already for practical cases, either symbolically generated equations or general multi body codes are the only practical way to generate reliable equations. A final judgment on the use of orthogonal shapes should be made in the context of that implementation.

Lagrange's equations applied by brute force to the appropriate energy functions will generate a complex dynamic model. Simplification can be achieved by simplification of the resulting complex equations can be pursued as described by Book[3]. Remarkable success has recently been achieved by prior simplification, using the specific form of the equations and the relation between the coefficients ultimately sought for the equations of motion.[8][9] Kinetic energy, for example, can be written as the integral over the link, but also as a quadratic product with the rate variables with the mass matrix.

$$KE = x^T M x^T/2 \qquad (6)$$

The nonlinear dynamic terms can be related to the changes of the mass matrix. Various relations like this have been used to represent rigid arm dynamics[10] and their analogies are now being found for flexible arms.

Non-serial arms

Very few of the thousands of manipulators constructed in the world qualify exactly as serial link manipulators. Only a pure direct drive arm can meet the qualification since all speed reductions involve parallel structural and drive elements. Incorporating parallel flexible links is difficult but manageable.[11] Differential equations can be formulated for each parallel path along the structure up to a point where they must connect. Algebraic constraint equations prescribing the meeting of the parallel paths must accompany the differential equations and their constraint forces. Several numerical techniques are then available to jointly solve these two types of equations and eliminate the extra degrees of freedom from the differential equations. Among the numerical procedures relevant are Singular Value Decomposition (SVD), QR decomposition, LU partitioning and Gaussian elimination.[11] SVD is attractive for flexible manipulators because the reduced variables resulting are tangent to

the constraint surface, and errors in satisfying the constraint equation have less effect on the overall dynamic simulation.

Symbolic Derivation and Multi-Body Codes

Hand derivation of dynamic equations for multi-link flexible arms is not recommended for producing the final equations of motion. It does give a student of the subject a healthy appreciation for the complexity of the problem and perhaps ideas for simplifying the result. Two alternatives are symbolic derivation and the use of general purpose multi-body codes.

General purpose symbolic manipulation programs include MACSYMA, SMP, REDUCE, and MAPLE. These were originally developed for main frame and mini computers but similar programs are now available on work stations and even personal computers. Special purpose symbolic codes are available for rigid arms (SDExact and SDFast) and are under development for similar flexible systems[12]. The general purpose symbolic codes allow one to approach research on more complex configurations with confidence. They cannot be viewed as an automatic means to turn theory into simulation code. Problems of practical complexity can swamp even large memories with "intermediate expression swell," especially when an expression is expanded in preparation to simplification. Pathological cases continue to be found on these systems which give incorrect results without even warning the user. They are extremely complex programs in general, and some have evolved over many years leading to both good and bad characteristics. While not a panacea, they are an invaluable, almost essential to one who would develop equations of motion for a flexible arm.

Guidance on the use of general purpose manipulation programs for rigid[13] and flexible manipulators using Lagrange's equations[14] is available in the literature. The special nature of kinematic chains leading to symmetry and zero terms can be exploited well with symbolic manipulation as shown in recent research.[8,9] for both serial and non serial arms.

Multi-body simulation codes are capable of handling unconstrained and in some cases constrained flexible chains. Well known codes include DISCOS, CONTOPS, DADS, ADAMS and others.[15] These codes insert the specifics of the model in numerical form early in the equation generation process. The disadvantage is more computational burden at each simulation time step. The advantage is a very general formulation suitable for components connected in a tree topology by various joints. They generally accept assumed shape data directly from finite element analysis modal results. Simplified linear models can be generated for more intensive design tradeoff studies on control, for example. In our research with flexible arms we have begun to look to these codes for independent verification of the accuracy of symbolically generated models.

Model order reduction

Discretization of the partial differential equations reduces the order of the flexible arm model from infinity to n. The value of n and the flexible degrees of freedom to be included can be determined from modal cost analysis techniques[16] that are most relevant to structures without feedback controls or to structures with the feedback controls already included. Tsujisawa[17] has applied this analysis to planar motion of a large arm with parallel actuated second link and found that one or two modes per flexible link were adequate for

his case. Large space structures may require dozens or hundreds of modes. The difference is the relatively clean, simple nature of the structure that Tsujisawa examined and that arms in general exhibit. Modal cost analysis does not insure that some higher mode will not cause instability i.e. that the effects of observation spill over will be small. This is especially problematical when the passive damping of the structure is small. Alberts[18] has shown the pronounced effects on stability of enhancing the passive damping through surface treatments. No general guarantee that a model includes the right degrees of freedom exists. A high order "truth model" for ultimate verification is perhaps the only insurance short of experiments.

Frequency Domain Analysis

It should be mentioned that if PDE descriptions of flexible elements are accurate and large motion behavior is not of interest, A very attractive alternative is frequency domain analysis. Models can be composed of elements which are flexible or rigid, serial or parallel, with up to six axes of freedom. Serial connection of components is much more readily incorporated using the transfer matrix approach. With the transfer matrix approach facilitates creation of the model from a library of elements as described in Book.[19] The principal price paid is a restriction to linear behaviors. While the powerful time domain synthesis techniques are not directly applicable to the frequency domain model, iterative techniques for pole placement were used by Book and Majette which converted between frequency domain and simple time domain models. Readily available are frequency response, natural frequency, true mode shapes, and, via the inverse FFT time response. The models have been applied to the Space Shuttle Manipulator Arm and a complex payload with general spatial motion. Numerical accuracy limited the approach as Majette used it, even with a 60 bit word length. More robust numerical techniques have been used in analysis of spinning spacecraft.[20] This approach is perhaps under utilized in the dynamics and control community. The powerful finite element techniques have dominated and seem more relevant to "messy problems" with many appendages and parallel connections.

2. DESIGN OF FLEXIBLE ARMS

This section is about the design of arms which behave well even though they are flexible. Designing arms to be flexible is not of practical interest.

Material properties principally affect strength, stiffness and damping. High strength materials allow lighter cross sections, consequently more flexibility. For many high performance materials, strength and stiffness increase together. While stiffness determines the need for flexible arm control algorithms, damping determines the ease in implementing such a control algorithm. Composite materials typically have more damping than homogeneous metals. Cost and ease of working with metals is a strong incentive for alternative means of damping enhancement.

The constrained layer damping treatment is a very effective means of enhancing damping for flexural vibrations.[18] One can achieve an increase in damping by a factor of 10. The treatment consists of a thin visco-elastic layer placed on the beam's surface and covered by a very stiff constraining layer. Bending of the beam results in shearing of the visco-elastic material and consequent dissipation of energy. This dissipation can be maximized for a given wavelength

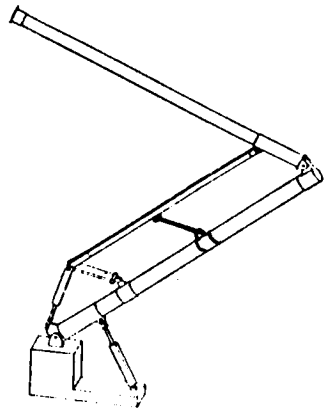


Figure 1: Robotic Arm Large and Flexible (RALF). Note multiple connections between parallel linkages.

of vibration by sectioning the constraining layer. The dissipation can be large for a wide range of wavelengths and their attendant frequencies, however. Torsional vibrations are not so readily coupled to the constrained layer treatment. A spiral wrap of a strip of constraining material has been examined by Dickerson[21] and was effective for this purpose. Vibrations with torsional compliance of a link depends on other links or the payload for inertia. These links would be at an angle to the link in torsion and hence undergo some flexure. This flexure could be effectively damped as mentioned before.

An active alternative to passive damping is active damping using piezo electric films or ceramics.[22] By closing the loop with local measurements of the strain, much the same effect of passive damping is obtained. One advantage of the active approach is the elimination of temperature sensitivity which is quite pronounced in the visco-elastic materials. Another advantage is a small amount of static deflection that can be obtained to control the shape of structures if that is important. The disadvantage is the complexity of an additional control loop and a high excitation voltage of several hundred volts.

Designs to minimize the flexibility of an arm are also important. Parallel link mechanisms are more rigid than a serial link mechanisms of equivalent weight such as the Stuart's platform. Unfortunately, the range of travel is typically smaller too. A parallel actuation link can have the dual benefits of placing motor mass near the base and providing a larger cross section area moment of inertia when the two parallel links are bent. Multiple connections between the parallel links such as shown in Fig. 0.1 work to further stiffen the pair by allowing both to support the load in bending. The added attraction of this arrangement is that the buckling load for the actuation link is increased.

Design should be interpreted broadly to include completely new concepts of arm operation. Additional degrees of freedom, e.g. a small arm, on the end of a long flexible arm are one way to change the nature of tradeoffs that must be made in arm design. The tradeoff between arm rigidity and low inertia can be couched in terms of gross and fine motion speeds. The extra degrees of freedom can be used to have light weight for gross motion of a large arm and high bandwidth rigid motion of a small arm it carries. These extra degrees of freedom can also be used to generate inertial forces that act to reduce vibration much as a dynamic vibration absorber would. They can also be used to

compensate for the relatively slow vibrations of a large arm, keeping the end point stationary. These three strategies are discussed in a companion paper, and will not be dealt with in detail here.

3. CONTROL FOR JOINTS OF FLEXIBLE ARMS

This section will discuss the limits for performance with rigid arm controls and the use of advanced control algorithms of various types. The control of only the existing joints will be discussed, not the addition of additional actuators specifically for controlling vibrations, such as proof masses, reaction wheels, or "smart materials." Such a control problem for robots is typically broken down into path planning, trajectory planning, and trajectory following. Little if any work on path planning specifically for flexible arm robots has been presented. The concentration will therefore be on the other two phases. For flexible robots control might also have the objective of vibration damping.

Trajectory Planning

Trajectory planning for the joint and the end point of a flexible arm are not equivalent problems as they are for the rigid arm. Three perspectives on the problem can be identified:

1. Generate an "optimal" trajectory
2. Control the arm's tip to follow a specified trajectory
3. Control the joints account for rigid link motion plus static deflection and suppress arm vibrations with feedback control.

Sangveraphunsiri[23] numerically determined the minimum time trajectory for a single link arm but recommended a near optimum based on rigid behavior and a feedback control near the final position due to the sensitivity of the true optimum to parameter variations. Meckl and Seering[24] solved a similar linearized problem in modal coordinates which yielded a simpler solution format applicable to the complete class of linear problems. Book and Cetinkunt[25] looked at trajectory optimization for rigid arms and extended it in an approximate way to flexible arms.

Singer and Seering[26] examined shaping techniques for trajectories in terms of their vibrational consequences. Work by Oosting and Dickerson[27] sought to choose tip trajectories to make the joint torques realizable without preview. A type of inverse dynamics model was used. Bayo[28] used a more elaborate model to obtain joint torque histories to track a Gaussian tip trajectory. Bayo's model was based on finite element and his original solution involved frequency domain techniques with substantial computational burden. He is extending his results to the two link case. These inverse dynamics approaches must simultaneously incorporate inverse kinematics, since the flexible degrees of freedom cannot be decoupled from the joint degrees of freedom. Asada also studied the inverse dynamics/kinematics problem and found that it was possible to produce a well behaved tip motion that resulted in unstable joint motion.[4] This was not observed by Bayo or Dickerson and may involve the choice of trajectories to be followed.

Several authors have planned the trajectory based on a static correction to the tip position based on rigid arm motion. Since this is not a dynamic correction it is easily implemented. Joint position adjustments assume instantaneous wind up of the arm "spring" predicted by acceleration forces. The method is shown to be very successful. It would seem that a good choice of trajectory to track is critical to the success of this method.

It should also be mentioned that if the tip position is not crucial except near the end of a motion, it is quite reasonable for the flexible arm to track joint motions based on rigid kinematics and use a final feedback control to produce acceptable vibration characteristics. When a flexible manipulator is used as a teleoperator, trajectories are at the disposal of the human operator, and can be at most minimally filtered to condition the reference signal.

Trajectory Tracking

As an arm is made lighter (more flexible) a point will be reached when it can no longer be considered rigid. Alternatively, if the arm servo bandwidth, ω_s is increased sufficiently, interaction of the dominant poles with the lowest ignored poles (in the rigid model) will eventually occur. For simple beams connected by rotary joints this lowest pole will be the first structural frequency with joints clamped, ω_c. The proximity of the closed loop bandwidth to ω_c gives one a valuable measure of the difficulty in achieving that bandwidth with simple rigid arm controllers. For example $\omega_s < \omega_c/2$ has been proposed as a practical limit for simple P.D. joint control.[29]

Imperfections in the system's behavior result in poor performance even for much stiffer systems. Coulomb friction in the joints, for example, will result in the link not back driving the actuator for small vibrations. These oscillations cannot be damped by the actuator motion, i.e. energy cannot be removed from the vibration. With greater frictional break away torque, larger amplitude vibrations will be allowed to exist with only structural damping slowly reducing their amplitude. With link strain included in the joint control, even low amplitude structural vibrations can be damped. For arms with speed reducers this is especially valuable, since some high ratio reducers are not back driveable under any circumstances. The strain feedback can be viewed either as a damping enhancement or as a inner torque control loop.

In order to account for a limited number of additional states in a flexible arm, a higher order model can be used in the control synthesis. Linear regulators or tracking controls optimized based on a quadratic performance index with a guaranteed margin of stability have been employed for end point[30] and strain measurements[31] for one link arms and also for multi link arms[32][8 Measurements involving link flexure can introduce non-minimal phase behavior. This is most apparent with tip position measurements of a one link arm. The tip initially moves in the opposite direction of the applied torque. The linear transfer function of such a system has zeros in the right half plane. The poles of an output feedback controller will move toward these zeros as gains increase, leading to instabilities. Viewed from a state space perspective, some optimization techniques for linear systems effectively cancel unwanted zeros with poles, leading to instability when the cancellation is inexact. Other measurements are less vulnerable to non minimum phase difficulties. Strain gages at the base of a one link arm with motor inertia has no non minimum phase zeros, and yet can be used to observe all system states.[33] The price paid is

the lack of direct knowledge about end point position in the uncertain work environment. For higher modes it is desirable to make more measurements instead of having a high order observer, with the attendant computational requirements. These additional sensors can introduce the non minimum phase zeros and instability. Multi link arms introduce an even greater need for additional measurements. Understanding more thoroughly the role of non minimum phase dynamics in both the linear and nonlinear cases is a very challenging and potentially useful research area. How can one combine strain and tip position sensors to achieve a robust and accurate controller?

The system linearity assumed in standard LQR control design is highly questionable in the robotic applications. Rigid arm control has been able to circumvent this through various means, including computed torque techniques, linearizing controllers, nonlinear controls (e.g. variable structure control), and adaptive controls. When the number of degrees of freedom exceeds the number of actuators, as for flexible arms, this approach must be modified.

Adaptive control has also been applied to the rigid case. Application of rigid arm model reference adaptive control to a flexible arm can overcome the adverse nonlinear forces at high velocities, but cannot overcome the bandwidth limitations imposed by vibrational structural modes.[34] For a flexible MRAC to be designed using the stability methods applied to rigid arms, the "model matching" condition must be satisfied. The rigid arm development uses the equal numbers of actuators and degrees of freedom. The near linearity of the one link flexible case has allowed Siciliano, et.al.[35] to accomplish model matching to a reference model which, instead of decoupling the rigid degrees of freedom, is a linearized model of the flexible arm. This also provides values of the flexible states for tracking during the motion. Others have proposed indirect adaptive control approaches such as the estimation of the payload mass.[36]

An explicit means of incorporating model uncertainty and simplifying assumptions is provided by the bounded uncertainty approach. Robustness can be enhanced by adding to the linear control adaptation and a saturation term similar to variable structure controls. This technique has been used to derive a decentralized controller for a two joint flexible arm with very good success relative to both a rigid controller and to a pure linear flexible feedback controller.[8]

The complexity of the nonlinear, flexible problem has lead researchers to seek various ways to simplify the problem. Rigid-nonlinear approximations are usual. Flexible-linear approaches are also common. When limited to small motions and inconsequential nonlinear velocity forces this is straight forward. Since the elastic deflections are usually rather small, linearization along a specified motion path is also effective. It is not accurate to assume the gross motions only force the flexible motions, since the damping of the vibrations is increased by their influence on the moving joint. One well developed approach for separating rigid and flexible motions is by exploiting their time scale separation with Singular Perturbation Analysis. This has been studied for arms with compliant drives[37] and with flexible links[38]. If flexible frequencies are to retain the broad separation from the "rigid body" frequencies needed for the singular perturbation theory to hold, high performance light weight arms will be automatically excluded. An alternative to including all flexible degrees of freedom in the fast system is to place the lowest mode in the slow system with the rigid body modes. This enables the dominant dynamics to remain

in the slow system. It appears feasible for light arms with significant payloads. For example, as the payload of a beam gets bigger and heavier, its first bending frequency approaches zero. The second bending frequency approaches the first clamped-clamped bending frequency. On a percentage basis, the separation of the two modes increases as the payload increases.

Practical advantage seems to be gained when the decoupling of rigid and flexible motion combines a linearizing feedback control for the rigid motion and a linear control on the flexible subsystem linearized about the rigid motion. A static deflection correction to the specified joint motion can also be incorporated to place the tip closer to its specified trajectory.[39]

4. CONCLUSIONS

The consideration of flexibility in manipulator arms is in a rapid state of progress relative to a few years ago, but much remains to be done. Many new approaches in modeling, design and control are being explored. It is important that experimentation accompany the theoretical and simulation results to keep realism in the research. Even single link experiments are much better than no experiments. It is important to move into realistic 2 and 3 joint experiments where flexibility is representative of real applications or at least scaled to those applications. It is possible that earlier work on rigid arms will find more application when the role of flexibility is more fully understood.

1. Chalhoub, Nabil G. and A. Galip Ulsoy, "Dynamic Simulation of a Leadscrew Driven Flexible Robot Arm and Controller," J. of Dynamic Systems, Measurement, and Control, vol.108, June, 1986, pp. 119-126.
2. Craig, R.R., Jr. and Bampton, M.C.C., "Coupling of Substructures for Dynamic Analysis," AIAA Journal, Vol 6, No. 7, 1968, pp. 1313-1319
3. Book, W.J., "Recursive Lagrangian Dynamics of Flexible Manipulators," The International Journal of Robotics Research,vol. 3, no. 3, pp.87-106, 1984.
4. Asada, H. and Z.-D. Ma, "Inverse Dynamics of Flexible Robot Arms," ASME Special Publication No. G00461, 1988.
5. Baruh, H. and S.S.K. Tadikonda, "Issues in the Dynamics and Control of Flexible Robot Manipulators," to appear in J. of Guidance, Control, and Dynamics.
6. Kane, T.R. and R.R. Ryan, "Dynamics of a Cantilever Beam Attached to a Moving Base," Journal of Guidance, Control and Dynamics, Vol. 10, 1987, pp. 139-151.
7. Kane, T.R. and D.A. Levinson, "Formulation of Equations of Motion for Complex Spacecraft," J. Guidance and Control, vol.3, no.2, pp. 99-112, 1980.
8. Yuan, B.-S. "Adaptive Strategies for Position and Force Controls of Flexible Arms" Ph.D. Thesis, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA., expected March, 1989.
9. Lee, J.-W., "Dynamic Analysis and Control of Lightweight Manipulators with Flexible Parallel Link Mechanisms," Ph.D. Thesis, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA., expected June, 1989.
10. Asada, H. and J-J.E. Slotine, Robot Analysis and Control,John Wiley & Sons, 1986.
11. Lee, Jae-Won, J.D. Huggins and W.J. Book, "Experimental Verification of a Large Flexible Manipulator," Proceedings, 1988 American Control Conference, June 15-17, 1988, Atlanta, GA, pp. 102-107.

12. Gluck, R., "Explicit Modeling and Concurrent Processing in the Simulation of Multibody Dynamic Systems," Proceedings of the Workshop on Multibody Simulation, G. Man and R. Laskin (eds.), April 15, 1988, Jet Propulsion Laboratory, JPL D-5190, pp. 983-1017
13. Koplik, J. and M.C. Leu, "Computer Generation of Robotic Dynamics Equations and Related Issues," J. of Robotic Systems, vol.3 no.3, 1986, pp. 301-319.
14. Cetinkunt, S. and Book, W. "Symbolic Modeling of Flexible Manipulators," Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, April, 1987, pp. 2074-2080.
15. Man, G. and R. Laskin (eds.), Proceedings of the Workshop on Multibody Simulation, April 15, 1988, Jet Propulsion Laboratory, JPL D-5190, pp. 478-690.
16. Skelton, R.E., R. Singh, and J Ramakrishnan, "Component Model Reduction by Component Cost Analysis," Proceedings, AIAA Guidance and Control Conf., Minneapolis, MN, Aug., 1988.
17. Tsujisawa, Takahiko and W.J. Book, "A Reduced Order Model Derivation for Lightweight Arms with a Parallel Mechanism" to appear, Proc., IEEE International Conference on Robotics and Automation, Scottsdale, AZ, May 14-19, 1989.
18. Alberts, T., S. Dickerson and W. Book, "Modeling and Control of Flexible Manipulators," Proc., ROBOTS 9 Exposition and Conference, Detroit, MI, June 2, 1985, pp. 1/59-1/73.
19. Book, W.J. and M. Majette, "Controller Design for Flexible Distributed Parameter Mechanical Arms via Combined State Space and Frequency Domain Techniques," J. of Dynamic Systems, Measurement and Control, Dec. 1983, pp. 245-249.
20. Poelaert, D., "Impedance Representation of Flexible Substructures," Proceedings of the Workshop on Multibody Simulation, G. Man and R. Laskin (eds.), April 15, 1988, Jet Propulsion Laboratory, JPL D-5190, pp. 318-353.
21. Dickerson, S.L. unpublished report on torsional constrained layer damping.
22. Bailey, T. and J.E. Hubbard, Jr., "Distributed Piezoelectric-Polymer Active Vibration Control of a Cantilever Beam," J. of Guidance, Control and Dynamics, Vol. 8, No. 5, pp. 605-611, 1985.
23. Book, W. and V. Sangveraphunsiri, "An Approach to the Minimum Time Control of a Simple Flexible Arm," Control of Manufacturing Processes and Robotics, W. Book, D. Hardt (ed.), ASME, December, 1983.
24. Meckl, P. and W. Seering, "Active Damping in a Three-Axis Robotic Manipulator," J. of Vibration, Acoustics, Stress, and Reliability in Design, Vol. 107, No. 4, Oct. 1985, pp. 38-46.
25. Book, W.J. and S. Cetinkunt, "Near Optimum Control of Flexible Robot Arms on Fixed Paths," 1985 IEEE Conference on Decision and Control, Ft. Lauderdale, FL, Dec 11-13, 1985.
26. Singer, N.C. and W.P. Seering, "Using Acausal Shaping Techniques to Reduce Robot Vibration," Proc. 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, April 24-29, 1988, pp. 1434-1439.
27. Oosting, K. and S.L. Dickerson, "Simulation of a High-Speed Lightweight Arm," Proc. 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, April 24-29, 1988, pp. 494-496.
28. Bayo, E., R. Movaghar and M. Medus, "Inverse Dynamics of a Single-Link Flexible Robot. Analytical and Experimental Results," International J. of Robotics and Automation, Vol. 3, No. 3, Fall, 1988.

29. Book, W.J., O. Maizza-Neto and D.E. Whitney, "Feedback Control of Two Beam, Two Joint Systems with Distributed Flexibility," J. Dynamic Systems, Measurement and Control, vol. 97G, no. 4., pp. 424-431, Dec. 1975.
30. Cannon, R.H. and E. Schmitz, "Initial Experiments on the End-Point Control of a Flexible One-Link Robot," The International J. of Robotics Research, vol.3, no. 3, 1984, pp. 62-75.
31. Hastings, G. and W. Book, "Experiments in Optimal Control of a Flexible Arm," Proceedings of the American Control Conference, June 1985, pp. 728-729.
32. Henrichfreise, H. W. Moritz and H. Siemensmeyer, "Control of a Light, Elastic Manipulation Device," Proceedings, Conference on Applied Motion Control 1987, Minneapolis, MN, June16-17, 1987, pp. 57-66.
33. Hastings, G. and W. Book, "Reconstruction and Robust Reduced-Order Observation of Flexible Variables," presented at the 1986 ASME Winter Annual Meeting, Annaheim, CA, Dec. 1986.
34. Cetinkunt, S. and W. Book, "Performance of Lightweight Manipulators under Joint Variable Feedback Control: Analytical Study of Limitations," Proc. 1988 American Control Conference, June 15-17, 1988, Atlanta, GA, pp. 1021-1028.
35. Siciliano, B., B.-S. Yuan, and W. Book, "Model Reference Adaptive Control of a One Link Flexible Arm," Proc. of the 25th IEEE Conference on Decision and Control, Athens, Greece, Dec. 10-12, 1986.
36. de Wit, C.C., and E. Van den Bossche, "Adaptive Control of a flexible arm with explicit estimation of the payload mass and friction." Preprints of the IFAC/IFIP/IMACS INT. SYMPOSIUM ON THEORY OF ROBOTS, P. Kopacek, I. Troch, and K Desoyer (eds.), Vienna, Austria, Dec. 3-5, 1985, pp. 315-320.
37. Spong, M.W., K. Khorasani and P.V. Kokotovic, "An Integral Manifold Approach to the Feedback Control of Flexible Joint Robots," IEEE J. Robotics and Automation, Vol. 3, No. 4, pp 291-300.
38. Siciliano, B. and W. Book, "A Singular Perturbation Approach to Control of Lightweight Flexible Manipulators," Internation Journal of Robotics Research, vol. 7, no. 4, pp. 79-90, August, 1988.
39. Pfeiffer, F., B. Gebler, and U. Kleemann, "On Dynamics and Control of Elastic Robots," Proceedings of Robot Control 1988 (SYROCO '88) an IFAC symposium, Karlsruhe, Fed. Rep. of Germany, Oct. 5-7, 1988.

Dynamical Modeling of Serial Manipulators with Flexible Links and Joints Using the Method of Kinematic Influence

Philip L. Graves
Lockheed Engineering and Sciences Company
Houston, Texas 77258

ABSTRACT:

A method of formulating the dynamical equations of a flexible, serial manipulator is presented, using the Method of Kinematic Influence. The resulting equations account for rigid body motion, structural motion due to link and joint flexibilities, and the coupling between these two motions. Nonlinear inertial loads are included in the equations. A finite order mode summation method is used to model flexibilities. The structural data may be obtained from experimental, finite element, or analytical methods. Nonlinear flexibilities may be included in the model.

INTRODUCTION:

Link and joint flexibility often have significant effects on the performance of robotic manipulators. Simulations which include the dynamical effects of flexibility should include the structural dynamics coupled with the dynamics due to the gross motion of the links. A method of formulating such a dynamical model is presented. It extends the Method of Kinematic Influence to include a finite order mode summation model of structural dynamics.

The Method of Kinematic Influence is used to obtain a geometric and kinematic description of the robotic device, which includes the effects of flexibility. The kinematic description is then used to obtain a dynamical model which includes structural motions, gross motion of the links and base, and the coupling terms between the structural and gross motions. Nonlinear inertial forces are included. The operations used in obtaining this model are simple transformations of the inertias of each link, and first order transformations of forces and torques. A computer program, called V-Sim, has been written which uses this method to automatically generate the dynamical model for simulations.

REVIEW OF PREVIOUS WORK:

Various models of structural dynamics in robots have been presented in the literature. Many of these models use a finite order modal representation of the distributed mass and stiffness of each link. [6-8,11,14-19,21,23-25] Some lumped parameter models, [1,5,13,20,22] and some finite element models [2,9,17] have been presented. Linearized and quasi-static models have also been analyzed to determine how they differ from the full non-linear dynamical model. [16,17,20]

The method used to derive the dynamical equations should be chosen because it is easy to understand, or because it meets some other desirable criteria. Lagrangian derivations are common in flexible body dynamical modeling because they use the kinetic and potential energies, which may be easily obtained for a system of flexible bodies. [6,13,23-25] Hamilton's Principle has also been used because it provides similar advantages. [17] Other derivations have used Newton's Laws. [7,20]

Many computational algorithms have been presented. Great variations exist in the order of the calculations, and in how the algorithms collect common operations and common terms. Recursive methods of computation have been popular because of their tractability and efficiency. [11] Other more general methods which do not depend upon a

specific recursive algorithm, have been presented. [3,4,10,12,19,23] Most methods may also be used to obtain the system inertia matrix and the non-linear torques which are necessary in control algorithms involving dynamical compensation.

The assumptions of the structural model have great effects on the validity of the resulting dynamical model. For example, mode summation models often assume modes to be geometrically decoupled at the local link level, and may ignore certain dynamical stiffening effects which may occur. [15,19] If inertial variations due to flexibility are included, these models will predict extremely large deflections and become unstable at high rotation rates. When such a model is used, the assumptions which restrict its application should be examined. In general, these models are valid only for small link deflections.

The Method of Kinematic Influence was developed first for rigid body, open loop (serial) mechanisms. [3,4,10] The method was extended to closed loop (parallel) mechanisms [12], and then to mechanisms with flexible joints [13,22], and flexible links [23]. This method is an extremely powerful and simple way of obtaining the dynamical model of a complex mechanism. Its organized structure yields information which is useful in mechanical design and analysis. It may also be used to calculate information about the system inertia matrix and non-linear forces and torques which are required in many advanced control algorithms.

THE GEOMETRY OF A FLEXIBLE SERIAL MECHANISM:

The geometry of a serial mechanism can be represented by a series of links connected by translational or rotational joints. A local coordinate frame is attached at the proximal end of each link. The z-axis coincides with the proximal joint axis. The x-axis is perpendicular to both the proximal joint and distal joint of the undeflected link. The undeflected link is represented by the vectors $\mathbf{a}_i$ and $\mathbf{s}_{i+1}$. The joint angle is denoted by ϕ_i, and the link angle by α_i, as show in figure 1. If the proximal joint is rotational, ϕ_i will be a variable, but if it is translational, s_i will be a variable. Link deflections are represented by a displacement vector, $\mathbf{d}$, and a small rotation vector, Θ.

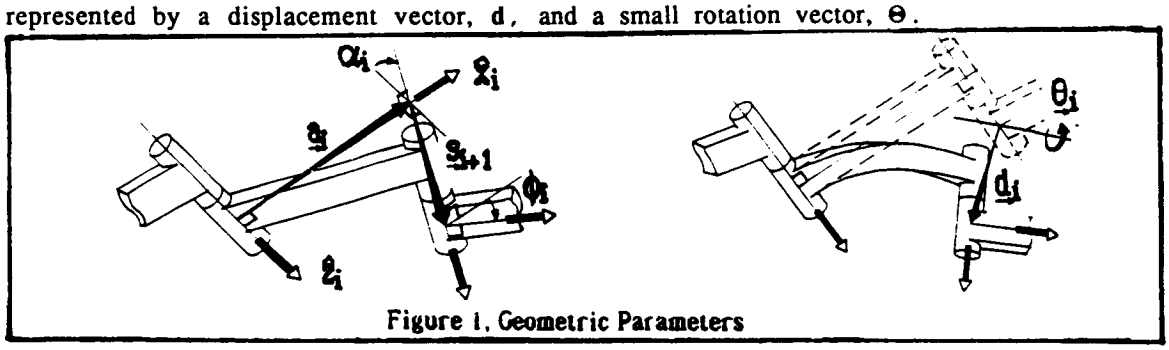


Figure 1. Geometric Parameters

A finite order modal representation is used to describe the structural deflections of each link. The deflection of a point on the link is a function of the magnitude of the modes of the mechanism, $\mathbf{q}$.

$$\begin{Bmatrix} \underline{d} \\ \underline{\theta} \end{Bmatrix}_p = \underset{\rightarrow}{\psi_p}(\underset{\rightarrow}{q}) = \begin{bmatrix} \psi_{tp}(\underset{\rightarrow}{q}) \\ \psi_{rp}(\underset{\rightarrow}{q}) \end{bmatrix} \tag{1}$$

Often, the modes are assumed to be geometrically decoupled at the local link level, and the deflections can be written in modal matrix form.

$$\begin{Bmatrix} \underset{\sim}{d} \\ \underset{\sim}{\theta} \end{Bmatrix}_p = \begin{bmatrix} \psi_p \end{bmatrix} \underset{\sim}{q} = \begin{bmatrix} \begin{bmatrix} \psi_{tp} \end{bmatrix} \\ \begin{bmatrix} \psi_{rp} \end{bmatrix} \end{bmatrix} \underset{\sim}{q} \tag{2}$$

The rotational coordinate transformation between sequential local coordinate frames can be represented in a 3x3 matrix form, $[{}^{i}T_{i+1}]$. The rotational deflection is represented by the skew-symmetric form of the small rotation vector, Θ_i, added to the identity matrix. This matrix is post-multiplied by a matrix representing the angle α_i about the x-axis, and then by a matrix representing the angle ϕ_i about the distal joint (z-axis).

$$[{}^{i}T_{i+1}] = \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Notice that the rotational transformation matrices between other frames may be found by concatenating these matrices, and that the inverse of a rotational transformation matrix may be approximated as the transpose, since the determinant is very close to one.

$$[{}^{h}T_i] = \prod_{j=h}^{i-1} [{}^{j}T_{j+1}] = [{}^{h}T_{h+1}][{}^{h+1}T_{h+2}] \cdots [{}^{i-1}T_i] \tag{4}$$

$$[{}^{h}T_i]^{-1} = [{}^{h}T_i]^{T} = [{}^{i}T_h] \tag{5}$$

The position vector of a point on link P is:

$$\underset{\sim}{R}_p = \sum_{j=h}^{p-1} \left\{ \underset{\sim}{a}_j + \underset{\sim}{d}_j + \underset{\sim}{s}_{j+1} \right\} + \underset{\sim}{x}_p + \underset{\sim}{d}_p \tag{6}$$

where x_p is the undeflected position of the point. All vectors are referenced to a common coordinate frame.

THE METHOD OF KINEMATIC INFLUENCE:

The Method of Kinematic Influence allows the cartesian velocities of any point on the mechanism to be expressed in terms of the positions and speeds of the joints, modes, and the base. This expression may be organized in the form of a Jacobian matrix, [J]. The translational and rotational cartesian velocities of the point, *P*, are described by a 6x1 vector, such that:

$$\begin{Bmatrix} \underset{\sim}{v} \\ \underset{\sim}{\omega} \end{Bmatrix}_p = [J_p] \begin{Bmatrix} \dot{\underset{\sim}{\phi}} \\ \dot{\underset{\sim}{q}} \end{Bmatrix} = \begin{bmatrix} \begin{bmatrix} J_{tp} \end{bmatrix} \\ \begin{bmatrix} J_{rp} \end{bmatrix} \end{bmatrix} \begin{Bmatrix} \dot{\underset{\sim}{\phi}} \\ \dot{\underset{\sim}{q}} \end{Bmatrix} \tag{7}$$

It is convenient to combine the joint variables with the modal variables in one vector. This hybrid combination of joint space and modal space will be called *j-m space*. Base motion is modeled as three rotational and three translational joints at the origin of the base link.

The columns of the Jacobian matrix are called the Kinematic Influence Coefficients, **g**, and are functions of the mechanism geometry, the joint positions, and the

deflections. For a rotational joint or base motion which contributes to the motion of point, P , the Kinematic Influence Coefficient is defined as:

$$\underset{\rightarrow}{g_i} = \begin{Bmatrix} \hat{\underset{\rightarrow}{S}}_i \times \underset{\rightarrow}{R}_{i/p} \\ \hat{\underset{\rightarrow}{S}}_i \end{Bmatrix} \tag{8}$$

For a translational joint or base motion which contributes to the motion of point, P , the coefficient is defined as:

$$\underset{\rightarrow}{g_i} = \begin{Bmatrix} \hat{\underset{\rightarrow}{S}}_i \\ \underset{\rightarrow}{0} \end{Bmatrix} \tag{9}$$

For a mode q_i which contributes to the motion of point P , the coefficient is defined as:

$$\underset{\rightarrow}{g_i} = \begin{Bmatrix} \left\{ \frac{\partial \underset{\rightarrow}{\psi}_{rp}}{\partial q_i} \right\} \times \underset{\rightarrow}{R}_{i/p} + \left\{ \frac{\partial \underset{\rightarrow}{\psi}_{tp}}{\partial q_i} \right\} \\ \left\{ \frac{\partial \underset{\rightarrow}{\psi}_{rp}}{\partial q_i} \right\} \end{Bmatrix} \tag{10}$$

But, if the modes are geometrically decoupled, the column of the modal matrix which is associated with this mode may be substituted for the partial derivatives.

$$\underset{\rightarrow}{g_i^p} = \begin{Bmatrix} \left\{ \left[\psi_{rp}^{'} \right]_{;i} \times \underset{\rightarrow}{R} \right\} + \left[\psi_{tp}^{'} \right]_{;i} \\ \left[\psi_{rp}^{'} \right]_{;i} \end{Bmatrix} \tag{11}$$

For a joint or mode which does <u>not</u> contribute to the motion of point, P , the coefficient is defined as:

$$\underset{\rightarrow}{g_i^p} = \underset{\rightarrow}{0} \tag{12}$$

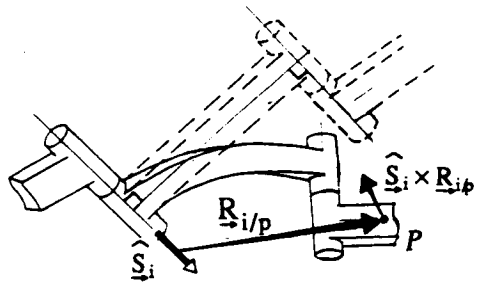


Figure 2. The Geometric Influence Coefficient for Joint i

The **S** and **R** vectors can be found from our knowledge of the geometry of the mechanism. **R** is the vector position from the joint or deflection to the point, P, and **S** is the vector of direction cosines which describe the line-of-action of the joint or deflection, as shown in Figure 2. All of the vectors which are used in these formulas must be expressed in a common frame of reference.

KINEMATIC INFLUENCE AND EQUIVALENT FORCES:

The Jacobian matrix also serves as a relationship between cartesian force/torques and the equivalent j-m loads:

$$[J_p]^T \begin{Bmatrix} \underset{\rightarrow}{F} \\ \underset{\rightarrow}{M} \end{Bmatrix}_p = \begin{Bmatrix} \underset{\rightarrow}{\tau} \\ \underset{\rightarrow}{Q} \end{Bmatrix} \tag{13}$$

where $\{\tau\ Q\}$ is the vector of equivalent loads in j-m space, $\{F\ M\}$ is the 6x1 vector of cartesian forces and torques, and $[J_p]^T$ is the transpose of the Jacobian Matrix for the point where $\{F\ M\}$ is applied. This relationship is a result of the duality of forces and velocities, and may be proven by showing that the virtual work performed by the cartesian force/torque is equal to the virtual work done by the equivalent j-m loads.

$$\begin{Bmatrix} \underset{\rightarrow}{F} \\ \underset{\rightarrow}{M} \end{Bmatrix}_p \cdot \begin{Bmatrix} \underset{\rightarrow}{\delta x} \\ \underset{\rightarrow}{\delta\theta} \end{Bmatrix}_p = \begin{Bmatrix} \underset{\rightarrow}{\tau} \\ \underset{\rightarrow}{Q} \end{Bmatrix} \cdot \begin{Bmatrix} \underset{\rightarrow}{\delta\phi} \\ \underset{\rightarrow}{\delta q} \end{Bmatrix} \tag{14}$$

THE DYNAMICAL EQUATIONS:

Consider a differential element of mass in one of the links. The kinetic energy of this mass element is:

$$KE_p = \frac{1}{2} \begin{Bmatrix} \underset{\rightarrow}{v_p} \\ \underset{\rightarrow}{\omega_p} \end{Bmatrix}^T \begin{bmatrix} \delta m_p[I] & [0] \\ [0] & [\delta\Pi_p] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{v_p} \\ \underset{\rightarrow}{\omega_p} \end{Bmatrix} \tag{15}$$

The velocities are referenced to a local coordinate frame fixed in the element. Potential Energy is defined as the integral of an elastic force and moment, from a reference position where there is no Potential Energy, to the current position, allowing for nonlinear stiffnesses in the system.

$$PE = \int_{\left(\underset{\rightarrow}{x},\ \underset{\rightarrow}{\Theta}\right)_{Reference}}^{\left(\underset{\rightarrow}{x},\ \underset{\rightarrow}{\Theta}\right)} \begin{Bmatrix} \underset{\rightarrow}{F_p} \\ \underset{\rightarrow}{M_p} \end{Bmatrix}_{Elastic} d\left(\underset{\rightarrow}{x},\ \underset{\rightarrow}{\Theta}\right) \tag{16}$$

Lagrange's Equation provides a convenient starting point for deriving the final form of the dynamical equations of the mass element.

$$\tau_j = \frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{x}_j}\right) - \frac{\partial KE}{\partial x_j} + \frac{\partial PE}{\partial x_j} \tag{17}$$

By chosing x to be the cartesian coordinates fixed in the mass element, the resulting equation is a familiar form.

$$\begin{Bmatrix} \underset{\rightarrow}{F_p} \\ \underset{\rightarrow}{M_p} \end{Bmatrix}_{External} = \begin{bmatrix} \delta m_p[I] & [0] \\ [0] & [\delta\Pi_p] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{a_p} \\ \underset{\rightarrow}{\alpha_p} \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{0} \\ \underset{\rightarrow}{\omega_p} \times [\delta\Pi_p]\ \underset{\rightarrow}{\omega_p} \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{F_p} \\ \underset{\rightarrow}{M_p} \end{Bmatrix}_{Elastic} \tag{18}$$

But this form of equation is not suitable for simulation. The equations must be converted to j-m space, and all of the other mass elements in the mechanism must be considered.

To express these dynamical equations in terms of j-m space, the Jacobian transpose relationship is used, where $[J_p]$ is the Jacobian for the mass element.

$$\begin{Bmatrix} \underset{\rightarrow}{T}_p \\ \underset{\rightarrow}{Q}_p \end{Bmatrix}_{External} = [J_p]^T \left\{ \begin{bmatrix} \delta m_p[I] & [0] \\ [0] & [\delta I_p] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{0} \\ \underset{\rightarrow}{\omega}_p \times [\delta I_p] \, \underset{\rightarrow}{\omega}_p \end{Bmatrix} \right\} + \begin{Bmatrix} \underset{\rightarrow}{T}_p \\ \underset{\rightarrow}{Q}_p \end{Bmatrix}_{Elastic} \tag{19}$$

Next, the dynamical equations for each mass element in the mechanism must be combined to form the dynamical equations for the system. This can be accomplished by adding all of the equations together, and can be written as a volumetric integral throughout the mechanism, assuming the mass elements are very small. Notice that the forces between the particles cancel because they are equal and opposite, <u>except</u> at the joints, and "at" the modes.

$$\begin{Bmatrix} \underset{\rightarrow}{T} \\ \underset{\rightarrow}{Q} \end{Bmatrix}_{External} = \int_V [J_p]^T \left\{ \begin{bmatrix} \rho_p[I] & [0] \\ [0] & \left[\frac{II}{V_p}\right] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{0} \\ \underset{\rightarrow}{\omega}_p \times \left[\frac{II}{V_p}\right] \underset{\rightarrow}{\omega}_p \end{Bmatrix} \right\} dV + \begin{Bmatrix} \underset{\rightarrow}{T} \\ \underset{\rightarrow}{Q} \end{Bmatrix}_{Elastic} \tag{20}$$

where ρ is the mass density and II/V is the inertia "density". For practical purposes, the integral term must be simplified. Let the acceleration of the particle can be expressed as the sum of a linear function of the j-m accelerations, and a nonlinear function of the j-m velocities.

$$\begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix} = [J_p] \begin{Bmatrix} \ddot{\phi} \\ \dot{q} \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix}_{Nonlinear} \tag{21}$$

Substituting this formula into the integral,

$$\int_V [J_p]^T \left\{ \begin{bmatrix} \rho_p[I] & [0] \\ [0] & \left[\frac{II}{V_p}\right] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix} + \begin{Bmatrix} \underset{\rightarrow}{0} \\ \underset{\rightarrow}{\omega}_p \times \left[\frac{II}{V_p}\right] \underset{\rightarrow}{\omega}_p \end{Bmatrix} \right\} dV$$

$$= \int_V [J_p]^T \left\{ \begin{bmatrix} \rho_p[I] & [0] \\ [0] & \left[\frac{II}{V_p}\right] \end{bmatrix} [J_p] \begin{Bmatrix} \ddot{\phi} \\ \dot{q} \end{Bmatrix} + \begin{bmatrix} \rho_p[I] & [0] \\ [0] & \left[\frac{II}{V_p}\right] \end{bmatrix} \begin{Bmatrix} \underset{\rightarrow}{a}_p \\ \underset{\rightarrow}{\alpha}_p \end{Bmatrix}_{Nonlinear} + \begin{Bmatrix} \underset{\rightarrow}{0} \\ \underset{\rightarrow}{\omega}_p \times \left[\frac{II}{V_p}\right] \underset{\rightarrow}{\omega}_p \end{Bmatrix} \right\} dV$$

$$= \int_V \left\{ [J_p]^T \begin{bmatrix} \rho_p[I] & [0] \\ [0] & \left[\frac{II}{V_p}\right] \end{bmatrix} [J_p] \right\} dV \begin{Bmatrix} \ddot{\phi} \\ \dot{q} \end{Bmatrix} + \int_V \begin{Bmatrix} \underset{\rightarrow}{T}_p \\ \underset{\rightarrow}{Q}_p \end{Bmatrix}_{Nonlinear} dV \tag{22}$$

The nonlinear inertial forces are lumped in one term for notational brevity. Notice that the linear inertial integral involves a similarity transformation. This will yield the system inertia matrix. It is most convenient to perform the integration over each link, and then sum the results. To do this, it is necessary to express $[J_p]$ in terms of the Jacobian of the link coordinate frame, and a Jacobian expressing the motion of the mass element with respect to the link coordinate frame.

$$[J_p] = \begin{bmatrix} [I] & \left[[\tilde{\underline{x}} + \tilde{\underline{d}}_p]^T\right] & [\psi_{p/i}] \\ [0] & [I] & [\psi_{r_{p/i}}] \end{bmatrix} \begin{bmatrix} \begin{bmatrix} [J_{t_i}] \\ [J_{r_i}] \end{bmatrix} \\ [0][I] \end{bmatrix} = [J_{p/i}] \begin{bmatrix} [J_i] \\ [0][I] \end{bmatrix} \tag{23}$$

The nxn zero matrix and mxm identity matrix in this formula are used to make the matrix multiplication conformable (n equals the number of joints, and m equals the number of modes of the mechanism). The matrix $[\psi_{p/i}]$ is equivalent to $[\psi_p] - [\psi_i]$. The linear inertial integral then becomes the definition of the system inertia matrix:

$$\int_V \left\{ [J_p]^T \begin{bmatrix} \rho_k[I] & [0] \\ [0] & \begin{bmatrix} \underline{\underline{\Pi}} \\ V_p \end{bmatrix} \end{bmatrix} [J_p] \right\} dV$$

$$= \sum_i \left\{ \begin{bmatrix} [J_i] \\ [0][I] \end{bmatrix}^T \left[\int_{V_i} [J_{p/i}]^T \begin{bmatrix} \rho_k[I] & [0] \\ [0] & \begin{bmatrix} \underline{\underline{\Pi}} \\ V_p \end{bmatrix} \end{bmatrix} [J_{p/i}] \, dV \right] \begin{bmatrix} [J_i] \\ [0][I] \end{bmatrix} \right\}$$

$$= \sum_i \left\{ \begin{bmatrix} [J_i] \\ [0][I] \end{bmatrix}^T [\Pi_i] \begin{bmatrix} [J_i] \\ [0][I] \end{bmatrix} \right\} = \sum_i [\Pi_i^*] = [\underline{\underline{\Pi}}^*] \tag{24}$$

If this integration is performed in the local link coordinate frame, and the modes are assumed to be decoupled at the local link level, and variations due to the flexibility are not considered, the elements can be defined as:

$$[{}^i\Pi_i] = \begin{bmatrix} [\Pi_{xx}] & [\Pi_{x\theta}] & [\Pi_{xq}] \\ [\Pi_{x\theta}]^T & [\Pi_{\theta\theta}] & [\Pi_{\theta q}] \\ [\Pi_{xq}]^T & [\Pi_{\theta q}]^T & [\Pi_{qq}] \end{bmatrix}_i \tag{25}$$

where:

$$[\Pi_{xx}] = m_i [I] \tag{25.a}$$

$$[\Pi_{x\theta}] = m_i [\widetilde{\underline{cm}}_i]^T = m_i \begin{bmatrix} 0 & cm_z & -cm_y \\ -cm_z & 0 & cm_x \\ cm_y & -cm_x & 0 \end{bmatrix} \tag{25.b}$$

$$[\Pi_{xq}] = \int_V \rho(\underset{\rightarrow}{x}) \begin{bmatrix} [\psi_i(\underset{\rightarrow}{x})] \\ [0] \end{bmatrix}^T dV \tag{25.c}$$

$$[\Pi_{\Theta\Theta}] = [\Pi_{cm_i}] + [\underset{\rightarrow}{cm}^T \underset{\rightarrow}{cm}\,[I] - \underset{\rightarrow}{cm}\,\underset{\rightarrow}{cm}^T] m_i \tag{25.d}$$

$$[\underset{\rightarrow}{cm}^T \underset{\rightarrow}{cm}\,[I] - \underset{\rightarrow}{cm}\,\underset{\rightarrow}{cm}^T] = \begin{bmatrix} cm_y^2+cm_z^2 & -cm_x cm_y & -cm_x cm_z \\ -cm_x cm_y & cm_x^2+cm_z^2 & -cm_y cm_z \\ -cm_x cm_z & -cm_y cm_z & cm_x^2+cm_y^2 \end{bmatrix} \tag{25.e}$$

$$[\Pi_{\Theta q}] = \int_V \begin{bmatrix} \rho(\underset{\rightarrow}{x})[\tilde{x}_i][\psi_i(\underset{\rightarrow}{x})] \\ \frac{I(\underset{\rightarrow}{x})}{V}[\psi_{r_i}(\underset{\rightarrow}{x})] \end{bmatrix} dV \tag{25.f}$$

and,

$$[\Pi_{qq}] = \int_V \begin{bmatrix} \rho(\underset{\rightarrow}{x})[\psi_i(\underset{\rightarrow}{x})]^T[\psi_i(\underset{\rightarrow}{x})] & [0] \\ [0] & \frac{I(\underset{\rightarrow}{x})}{V}[\psi_{r_i}(\underset{\rightarrow}{x})]^T[\psi_{r_i}(\underset{\rightarrow}{x})] \end{bmatrix} dV \tag{25.g}$$

such that $[\Pi_{cm}]$ is the rigid body rotational inertia at the center of mass, m_i is the total link mass, and $\underset{\rightarrow}{cm}$ is the undeflected position of the center of mass in the link coordinate frame. In practical situations, these inertial parameters may be estimated via finite element analysis, or some appropriate experimental technique. To transform the inertial parameters back into a common frame (which is necessary), the matrix is pre- and post-multiplied by a transformation matrix, where the mxm identity matrix is used to make the matrix multiplication conformable.

$$[\Pi_i] = \begin{bmatrix} [{}^hT_i] & [0] & [0] \\ [0] & [{}^hT_i] & [0] \\ [0] & [0] & [I] \end{bmatrix} [{}^i\Pi_i] \begin{bmatrix} [{}^hT_i] & [0] & [0] \\ [0] & [{}^hT_i] & [0] \\ [0] & [0] & [I] \end{bmatrix}^T \tag{26}$$

Often the nonlinear inertial terms are presented as Christoffel Symbols of the inertia matrix, which are multiplied by the appropriate joint velocities to obtain the nonlinear loads. The computation involved in computing the Christoffel Symbols is overwhelming for a mechanism with many flexible modes, and the mathematical operations involved are not easy to understand. The number of computations can be minimized by collecting common operations. To do so, the nonlinear acceleration of each link is computed using an iterative algorithm, then the nonlinear loads on each link are computed, and finally the loads are transformed back into j-m space using the $[J^T]$ relationship. The nonlinear loads will be computed from the accelerations and angular velocities of the center of mass of the link. The nonlinear accelerations may be computed in an iterative fashion:

$$\begin{Bmatrix} \underset{\rightarrow}{a} \\ \underset{\rightarrow}{\alpha} \end{Bmatrix}_{nonlinear_b} = \begin{Bmatrix} \underset{\rightarrow}{a} \\ \underset{\rightarrow}{\alpha} \end{Bmatrix}_{nonlinear_a} + \begin{Bmatrix} \underset{\rightarrow}{\omega_a} \times \underset{\rightarrow}{V_{b/a}} \\ \underset{\rightarrow}{\omega_a} \times \underset{\rightarrow}{\omega_b} \end{Bmatrix} \tag{27}$$

And the nonlinear forces are approximated by:

$$\begin{Bmatrix} \underset{\sim}{T} \\ \underset{\sim}{Q} \end{Bmatrix}_{\text{Nonlinear}} = \sum_i [J_{cm_i}]^T \left\{ \begin{bmatrix} m_i[I] & [0] \\ [0] & [I_{c_i}] \end{bmatrix} \begin{Bmatrix} \underset{\sim}{a} \\ \underset{\sim}{\alpha} \end{Bmatrix}_{c_i,\,\text{Nonlinear}} + \begin{Bmatrix} \underset{\sim}{0} \\ \underset{\sim}{\omega}_{c_i} \times [I_{c_i}]\,\underset{\sim}{\omega}_{c_i} \end{Bmatrix} \right\}$$

The final dynamical equations are expressed in j-m space, and in a standard form:

$$\begin{Bmatrix} \underset{\sim}{T} \\ \underset{\sim}{Q} \end{Bmatrix}_{\text{Applied}} = [\Pi^*] \begin{Bmatrix} \underset{\sim}{\ddot{\phi}} \\ \underset{\sim}{\ddot{q}} \end{Bmatrix} + \begin{Bmatrix} \underset{\sim}{T} \\ \underset{\sim}{Q} \end{Bmatrix}_{\text{Nonlinear}} + \begin{Bmatrix} \underset{\sim}{T} \\ \underset{\sim}{Q} \end{Bmatrix}_{\text{Elastic}} \tag{28}$$

where $[\Pi^*]$ is the system inertia matrix in j-m space, and the vectors of externally applied, and nonlinear inertial loads are given in j-m space.

V-Sim: A SIMULATOR FOR FLEXIBLE ROBOTICS:

These dynamical equations have been implemented as a computer program named V-Sim. It is currently being used in a variety of applications ranging from simulation of cantilever beams to simulations of the Space Shuttle Remote Manipulator System. The program automatically formulates the dynamical model for an open-loop manipulator. The manipulator may have n joints, which may be translational and rotational, and may have m modes of vibration. The resulting equations of motion may be used in simulations for controls design and analysis, mechanical design and analysis, or operational assessments.

REFERENCES:

[1] Benedict, C. E., Tesar, D., "Dynamic Response Analysis of Quasi-Static Mechanical Systems Using Kinematic Influence Coefficients", *ASME Journal of Mechanisms*, Vol. 6, 1971, pp. 383-403.

[2] Dubowsky, S., Gardner, T. N., "Design and Analysis of Multilink Flexible Mechanisms With Multiple Clearance Connections," *ASME Journal of Engineering for Industry*, February 1977, pp. 88-96.

[3] Benedict, C. E., Tesar, D., "Model Formulation of Complex Mechanisms with Multiple Inputs: Part I - Geometry," *ASME Journal of Mechanical Design*, October 1978, pp. 747-750.

[4] Benedict, C. E., Tesar, D., "Model Formulation of Complex Mechanisms with Multiple Inputs: Part II - The Dynamic Model," *ASME Journal of Mechanical Design*, October 1978, pp. 751-761.

[5] Sanders, J. R., Tesar, D., "The Analytical and Experimental Evaluation of Vibratory Oscillations in Realistically Proportioned Mechanisms," *ASME Journal of Mechanical Design*, October 1978, pp. 762-768.

[6] Book, Wayne J., "Analysis of Massless Elastic Chains With Servo Controlled Joints," *ASME Journal of Dynamic Systems, Measurement, and Control*, September 1979, pp 187-192.

[7] Hughes, P. C., "Dynamics of a Chain of Flexible Bodies", *Journal of the Astronautical Sciences*, Oct.-Dec. 1979, pp. 359-380.

[8] Huston, R. L., "Multi-Body Dynamics Including the Effects of Flexibility and Compliance," *Computers and Structures*, Vol. 14, No. 5-6, 1981, pp 443-451.

[9] Sunada, W., Dubowsky, S., "The Application of Finite Element Methods to the Dynamic Analysis of Spatial and Co-planar Linkage Systems," *ASME Journal of Mechanical Design*, July 1981, pp 643-651.

[10] Thomas, M., Tesar, D., "Dynamic Modeling of Serial Manipulator Arms," *ASME Journal of Dynamic Systems, Measurement, and Control*, September 1982, pp. 218-228.

[11] Book, W. J., "Recursive Lagrangian Dynamics of Flexible Manipulators", *Internalional Journal of Robotics Research*, Fall 1984, pp. 87-101.
[13] Freeman, R. , "Kinematic and Dynamic Modeling, Analysis, and Control of Robotic Mechanisms (Via Generalized Coordinate Transformation), Ph.D. Dissertation, University of Florida, Gainesville, August 1985.
[14] Behi, F., "Dynamic Analysis and Parametric Identification for Flexible Serial Manipulators," Ph.D. Dissertation, The University of Florida, Gainsville, 1985.
[15] Centinkunt, S., Siciliano, B., Book, W. J., "Symbolic Modeling and Dynamic Analysis of Flexible Manipulators", *Proc. 1986 IEEE International Conference on Systems, Man, and Cybernetics*, October 1986, pp. 798-803.
[16] Eke, Fidelis O., Laskin, Robert A., "On the Inadequacies of Current Multi-Flexible Body Simulation Codes," *Proc. 1987 AIAA Guidance and Control Conf.*, 1987, pp. 79-89.
[17] Low, K. H., "A Systematic Formulation fo Dynamic Equations for Robot Manipulators with Elastic Links," *Journal of Robotic Systems*, Vol. 4, No. 3, 1987, pp. 436-456.
[18] Naganathan, G., Soni, A. H., "Coupling Effects of Kinematics and Flexibility in Manipulators,"*The International J. Robotics Research*, Spring 1987, pp. 75-84.
[19] Hastings, Gordon G., Book, W. J., "A Linear Dynamic Model for Flexible Robotic Manipulators," *IEEE Control Systems Magazine*, February 1987, pp. 61-64.
[20] Kane, T. R., Ryan, R. R., Banerjee, A. K., "Dynamics of a Cantilever Beam attached to a Moving Base," *AIAA J. Guidance* , March-April 1987, pp. 139-151.
[21] Huang, Y., Lee, C. S. G., "Generalization of Newton-Euler Formulation of Dynamic Equations to Nonrigid Manipulators," *Proceedings of the 1987 American Control Conference*, Vol. 1, June 1987, pp 72-77.
[22] King, J. O., Gourishakar, V. G., Rink, R. E., "Lagrangian Dynamics of Flexible Manipulators Using Angular Velocities Instead of Transformation Matricies," *IEEE Transactions on Systems, Man, and Cybernetics*, November-December 1987, pp. 1059-1068.
[23] Fresonke, D. A., Hernandez, E., Tesar, D., "Deflection Prediction for Serial Manipulators," *Proceedings 1988 IEEE International Conference on Robotics and Automation*, 1988, pp 482-487.
[24] Graves, P. L., "The Effect of Inertial Coupling in the Dynamics and Control of Flexible Robotic Manipulators," Masters Thesis, The University of Texas at Austin, May 1988.
[25] Low, K. H., Vidyasagar, M., "A Lagrangian Formulation of the Dynamic Model for Flexible Manipulator Systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, June 1988, pp. 175-181.
[26] Everett, L. J., "An Alternative Algebra for Deriving Equations of Motion of Flexible Manipulators," *Journal of Robotic Systems*, Vol. 5, No. 6, 1988, pp. 553-566.

CAPTURE OF FREE-FLYING PAYLOADS WITH FLEXIBLE SPACE MANIPULATORS

T.Komatsu, M.Uenohara, S.Iikura
R&D Center, Toshiba Corporation
Kawasaki, Japan

H.Miura, I.Shimoyama
The University of Tokyo
Tokyo, Japan

Abstract

The purpose of this paper is to discuss a recently developed control system for capturing free-flying payloads with flexible manipulators. Three essential points in this control system are, calculating optimal path, using a vision sensor for an external sensor, and active vibration control. Experimental results are shown using a planar flexible manipulator.

1. Introduction

In the near future, capture of free-flying payloads, for example, recovery of satellites, will become one of the most important tasks for manipulators on the space station. This task of space manipulators are different from those on the earth. Manipulators have to capture payloads without impact. Most space manipulators should be structurally flexible, reflecting the necessity for their light-weight based upon minimum energy comsumption and shipping cost, as well as handling of large mass payloads in a no gravity environment. Therefore, it is also necessary to control structural vibration in these space manipulators to accomplish this task, especially after capturing satellites. The purpose of this paper is to discuss a recently developed control system for capturing free-flying payloads with flexible manipulators. Experimental results are shown using a planar flexible manipulator.

Three essential points in this control system are, (1)calculating optimal path of manipulator tip, (2)direct sensing of position and attitude of a payload using a hand eye camera, and (3)active vibration control of a manipulator. Determining the optimal path is necessary to accomplish non impact capture of a payload. This path is obtained by minimizing the performance index which consists of relative position error, velocity error and acceleration of manipulator tip. Direct sensing of the relative position between a manipulator and a payload is necessary to move the manipulator tip precisely along the optimal path. The authors used a hand eye CCD camera and LED target marker for this sensor. Active vibration control is necessary for precise and quick control of a manipulator. A local PD feedback of joint angle and torque was used for robust positioning and vibration control.

An experimental equipment was built to investigate the validity of this control system. This equipment consists of an air suspended, flexible two-link manipulator, a payload, and a controller. Experimental results of capturing a moving payload demonstrates the effectiveness of this system.

2. Control System

The task of a manipulator considered in this paper is automatic capture

of free-flying payloads, for example, capture of satellites, with manipulators on the space station or on the free-flyer. This task was accomplished in the Space Shuttle program by cooperation between RMS (remote manipulator system) and EVA (extra vehicular activities). This task was done for repair of a disabled satellite. In the future, after constructing the space station, many satellites will be assembled on the station and thrown into an orbit. Therefore, maintenance and repair of satellites will be done frequently on an orbit. In such a case, doing this task manually will be expensive and uneffective. Therefore accomplishing this task automatically is necessary.

To construct control system for the first stage, the following assumptions were made for payload motion.

(1)A payload rests or moves with constant velocity to the manipulator's base.
(2)A payload does not revolve on its own axis.
(3)Movement of a payload is constrained on one plane.

For using this control system on an orbit, the following capabilities are targeted:

(a)Real time manipulator path planning
(b)Direct sensing of the payload position
(c)Quick and precise motion control of a manipulator
(d)Easy integration

The system can be represented by Figure 1. The major subsystems are:

(1)Path planning
(2)Vision system
(3)Manipulator control

Explanation of the subsystem is as follows:

2.1 Path planning

In this subsystem, an approach path of a manipulator is calculated in real time. The following assumptions were made:
(1)A free-flying payload moves straight with constant velocity.
(2)The position and velocity of calculated path become equal to those of free-flying payload at the target time t_f.

The authors chose the optimal path obtained by minimizing the following performance index as the approach path satisfying upper assumptions.

$$J=\frac{1}{2}(\mathbf{X}^t S_f \mathbf{X})_{tf}+\frac{1}{2}\int_{t0}^{tf} \mathbf{a}_m{}^t \mathbf{a}_m dt \qquad (1)$$

where

$$\mathbf{\kappa}=\mathbf{\kappa}_p-\mathbf{\kappa}_m$$

$\mathbf{\kappa}_p$: The position vector of a free-flying payload
$\mathbf{\kappa}_m$: The position vector of a manipulator path

$$X=\begin{bmatrix}\dot{\mathbb{K}}\\ \mathbb{K}\end{bmatrix}=(Vx,\ Vy,\ Vz,\ X,\ Y,\ Z)^{t}$$

$$S_f=diag(C_1,\ C_1,\ C_1,\ C_2,\ C_2,\ C_2) \qquad (C_1,\ C_2\rightarrow\infty)$$

$$a_m=\ddot{\mathbb{K}}_m$$

By using this path, non impact capture is realized because position error and velocity error between a manipulator and a payload become zero at t_f. Moreover, smooth path is obtained because of minimizing total sum of $(a_m)^2$.

Considering a_m as the input to the system, the following system equation is obtained because the acceleration of a payload is zero.

$$\dot{X}=F\cdot X+G\cdot a_m \tag{2}$$

where

$$F=\left[\begin{array}{c|c} 0 & 0 \\ \hline 1\ 0\ 0 & \\ 0\ 1\ 0 & 0 \\ 0\ 0\ 1 & \end{array}\right] \qquad G=\left[\begin{array}{c} -1\ \ 0\ \ 0 \\ 0\ -1\ \ 0 \\ 0\ \ 0\ -1 \\ \hline 0 \end{array}\right]$$

Therefore, solving optimal path from equation (1) is equal to the design of the linear inhomogeneous regulator. In general, for this problem, the system equation, the cost function and its general solution become as follows [1]:

$$\dot{y}=F(t)\,y+G(t)\,u+C(t) \tag{3}$$

$$J=\frac{1}{2}(y^{t}S_f y)_{tf}+\frac{1}{2}\int_{t0}^{tf}(y^{t}Ay+u^{t}Bu)\,dt \tag{4}$$

$$u(t)=-B^{-1}G^{t}Sy-B^{-1}G^{t}K \tag{5}$$

$$\dot{S}=-SF-F^{t}S+SGB^{-1}G^{t}S-A \tag{6}$$

$$S_f=S(tf)$$

$$\dot{K}+(F^{t}-SGB^{-1}G^{t})K+SC=0 \tag{7}$$

$$K(tf)=0$$

Compairing equation (1) and (2) with (3) and (4),F and G equal equation (2), and y,u,B,A,C become as follows:

$$y=X \qquad u=a_m \tag{8}$$

$$B=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \qquad A=C=0.$$

Substituting these equations into equation (5)$\sim$(7), the analytical solution is obtained as follows:

$$\begin{bmatrix}a_{mX}\\a_{mY}\\a_{mz}\end{bmatrix}=\begin{bmatrix}\frac{4}{(tf-t)}V_X+\frac{6}{(tf-t)^2}X\\\frac{4}{(tf-t)}V_Y+\frac{6}{(tf-t)^2}Y\\\frac{4}{(tf-t)}V_Z+\frac{6}{(tf-t)^2}Z\end{bmatrix} \tag{9}$$

The optimal path is calculated by the integral of equation (9) substituting V_x, V_y, V_z, X, Y, Z which are obtained from vision system in real time.

2.2 Vision system

In this system, position and attitude data of a payload are sensed in real time. This system consists of a CCD camera, a marker and an image processing unit. A CCD camera watches a target marker on a payload. This marker consists of a rectangle formed by 4 LED points. Next, video signal from a CCD camera is sent to image processing unit. In this unit, coordinates of 4 LED points in the frame are detected using a simple hardware.

In three dimensional space, when the geometric relation of 4 points on a plane are known and corresponding points to 4 points are obtained in the frame, three dimensional coordinates of these points are determined by using inverse translation of projection. Therefore, three dimensional coordinates of a payload can be detected only by a monocular camera. Figure 2 shows coordinates of a camera$_f$(Σ C) which consist of X^c, Y^c and Z^c, and a frame which consist of X^f, Y^f. For each 4 LED points, the following equations are obtained from translation of projection.

$$\begin{bmatrix}X_i^c\\Y_i^c\\Z_i^c\end{bmatrix}=k_i\begin{bmatrix}X_i^f\\y_a\\Y_i^f\end{bmatrix} \qquad (i=1,\cdots,4) \tag{10}$$

where

$$k_1=\sqrt{\frac{L_{13}^2}{(X_1^f-k'_3\ X_3^f)^2+(Y_1^f-k'_3\ Y_3^f)^2+(1-k'_3\ )^2 y_a^2}} \tag{11}$$

$$\begin{bmatrix} k'_2 \\ k'_3 \\ k'_4 \end{bmatrix} = \begin{bmatrix} {X_2}^f & -{X_3}^f & {X_4}^f \\ y_a & -y_a & y_a \\ {Y_2}^f & -{Y_3}^f & {Y_4}^f \end{bmatrix}^{-1} \begin{bmatrix} {X_1}^f \\ y_a \\ {Y_1}^f \end{bmatrix} \qquad k_i = k_1 k'_i \quad (i=2, 3, 4)$$

and

$$\phi = \tan^{-1}\left(\frac{{X_1}^c - {X_4}^c}{{Z_1}^c - {Z_4}^c}\right) \qquad \theta = \cos^{-1}\left(\frac{{Z_1}^c - {Z_4}^c}{L_{14}\cos\phi}\right) \qquad (12)$$

$$\psi = \cos^{-1}\left[\frac{1}{L_{12}}\{({X_2}^c - {X_1}^c)\cos\phi - ({Z_2}^c - {Z_1}^c)\sin\phi\}\right]$$

ϕ, θ, ψ denote role, pitch, yaw of a marker. L_{ij} denotes the actual length between No. i and No. j. Therefore, three dimensional data ${X^c}$, ${Y^c}$, ${Z^c}$, ϕ, θ, ψ of a marker are detected by sensing ${X^f}$ and ${Y^f}$. In this system, data are obtained at 50 msec intervals.

2.3 Manipulator control

Most of space robots must be structurally flexible, reflecting the necessity for their light-weight based upon minimum energy consumption and shipping cost, as well as handling large mass payloads in a no gravity environment. Therefore it is necessary to control the structural vibration in the flexible arm for quick, precise tracking of the trajectories and accomplishment of tasks.

Figure 3 shows a blockdiagram of manipulator control. The outer loop is for motion control. The inner loop is for vibration control. In this method, joint angle and torque sensors are basically needed, which are ordinarily used in remote manipulator systems for nuclear power plants.

We derived this method from the standpoint of energy control in the system, where the energy was both kinetic and potential. We considered a flexible manipulator system as a completely mechanical system. From this standpoint, it is possible to consider that the control of a flexible manipulator is the same as the control of two types of mechanical springs in a system. The first one is springs in an elastic link. These springs cause structural vibrations of all modes when the accumulated potential energy converted to kinetic energy. So it is necessary to minimize both kinetic and potential energy at the target point for vibration restraint.

The second one is a spring in an actuator. For example, if the manipulator motion is controlled by simple proportional and derivative (PD) feedback compensation, as the general industrial robots, the work accomplished by this actuator is the same as the work in a general linear spring. In this case, the proportional control gain equals the spring constant. It is also necessary to minimize both the kinetic and potential energies of this spring at the target point for motion control [2].

In the case of flexible manipulators, both vibration and motion control are needed at the same time. Therefore, it is necessary to control both springs. In this control, total energy control for the system is accomplished by composing both joint angle and torque PD feedback loops. The position feedback loop realizes grobal motion control, while the joint torque feedback loop realizes vibration control of all modes [3].

3. Experimental setup

An experimental equipment was built to investigate the validity of this system. The authors named this equipment TESRA-I (teleoperated elastic space robot arm). This equipment consists of a two dimensional air suspended flexible manipulator, a payload and a controller. Figure 4 shows configuration of this equipment.

The flexible manipulator is about 1.5 meter long. It has two flexible links and three degrees of freedom (shoulder, elbow, wrist). An actuator is installed at each joint. It consists of a DC motor and a planetary gear reducer (1:100 reducer ratio). The sensor system consists of a potentiometer for sensing the joint angle, a tachogenerator for sensing the motor velocity, and the strain gages at the joint axis for sensing the joint torque. Flexible links for this manipulator are made form strainless steel. The link diameter is 6mm. The total weight for each joint and hand are 4kg and 1kg. This arm floats on an acrylic plate base, using four air bearings so as to simulate a no gravity environment in the horizontal plane. A small CCD camera, 35mm(W) x 43mm(H) x 70mm(L), is installed on the manipulator's hand (Fig.5).

The payload consists of lead sheets, and the weight is 40kg. A handle for grasping is installed at one side of this payload and a target marker is attached on it. This marker consisted of a rectangle formed by 4LED points, 40mm(W) × 30mm(H).

This manipulator is controlled by a MOTOROLA digital computer VME-10 system as the main computer. Its MPU is the 16-bit 68010, and the VERSAdos multitasking system is used as the operating system. Sensor outputs are sampled at 15msec intervals through a 32ch A/D board. Commands are fed to the servo amplifiers through the 4ch D/A board. In the vision system, 32-bit MPU(68020) are used for calculation of the target position.

4.Experimental result

Figure 6 shows one of experimental results of capturing a free-flying payload, and the paths of manipulator tip and a payload are indicated.

The task sequence from capture to stop is as follows. The time of capture t_f is 6.0sec. The path are planned so that manipulator tip goes to the location of about 1cm from a handle for grasping a payload. And, after arriving at the location, manipulator tip goes ahead and captures a payload.

(1)A payload is manually accelerated to get to about 5cm/sec speed, and released.
(2)An optimal path is calculated, and a manipulator is controled so as to track the planned path.
(3)A manipulator tip arrives at the location of about 1cm from a handle for grasping a payload at 6.0 seconds.

(4)A manipulator tip goes ahead for grasping a handle.
(5)Capturing of a payload finishes at 10 seconds.
(6)A manipulator stops a payload at 30 seconds.

It is obvious that a manipulator tip moved along a smooth path and captured a payload without impact. A little impact shown in figure 6 was caused when two fingers of a manipulator closed. No link vibration occured after stopping the payload.

5. Conclusions

In this study, the authors proposed a recently developed control system for use in automatic capturing free-flying payloads with a flexible manipulator. The features of this system are as follows:

(1)real time path planning
(2)direct sesing of payload position by a vision system
(3)quick and precise control of a flexible manipulator
(4)easy integration.

The effectiveness of this system has been verified by experimental results. The next step for this sytem is to expand its ability so as to adapt it to capture a payload with acceleration and revolution on its own axis, and finally adapt it to real three dimensional system.

References

1. V.Carber: Optimal Intercept Laws for Accelerating Targets, AIAA Journal, No.6, Vol.11, 6-11, (1968)

2. M.Takegaki and S. Arimoto: A New Feedback Method for Dynamic Control of Manipulators, ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 102, 119-125, (1981)

3. T.Komatsu, M.Uenohara, S.Iikura, H.Miura and I.Shimoyama: Active Vibration Control for Flexible Space Environment Use Manipulator, IFAC, Zurich, Switzerland (1988)

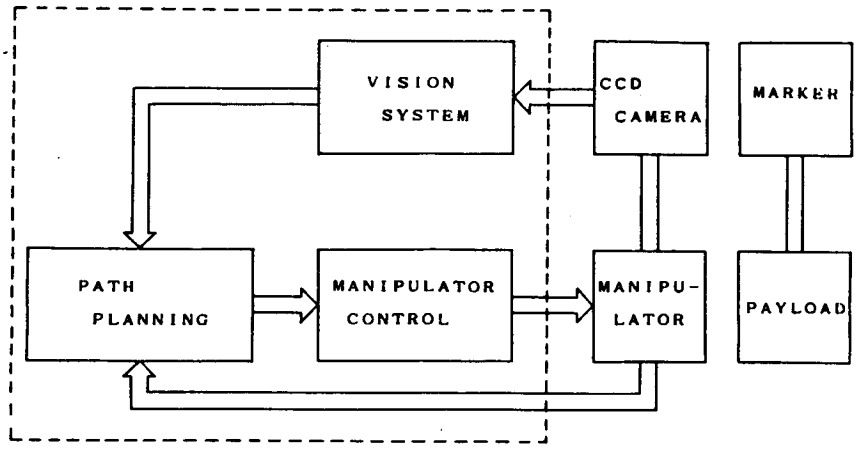


Fig. 1 Control System Architecture

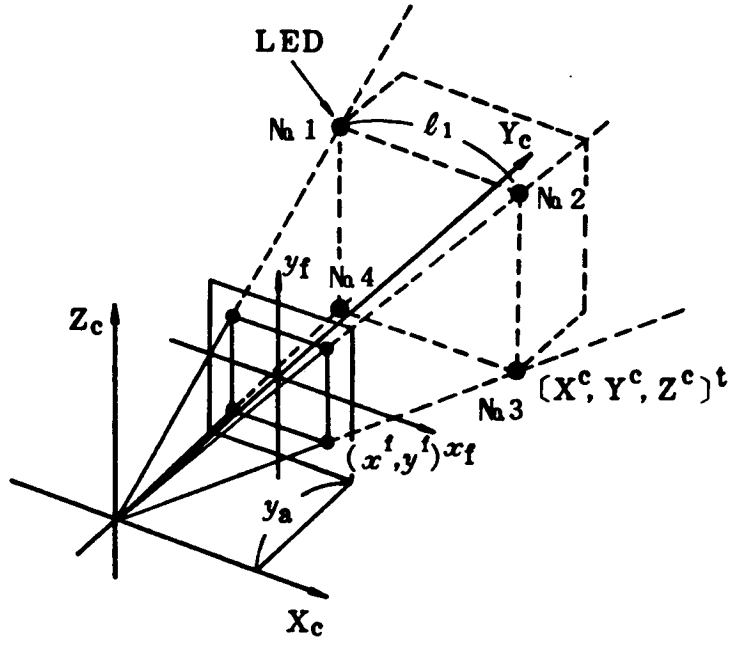


Fig. 2 Translation of Projection

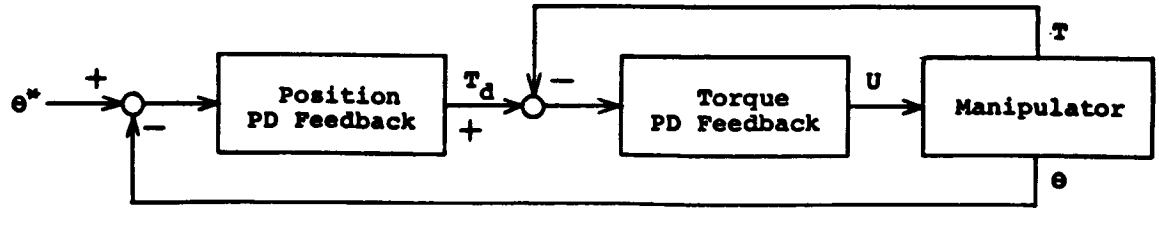


Fig. 3 Block diagram of Manipulator Control

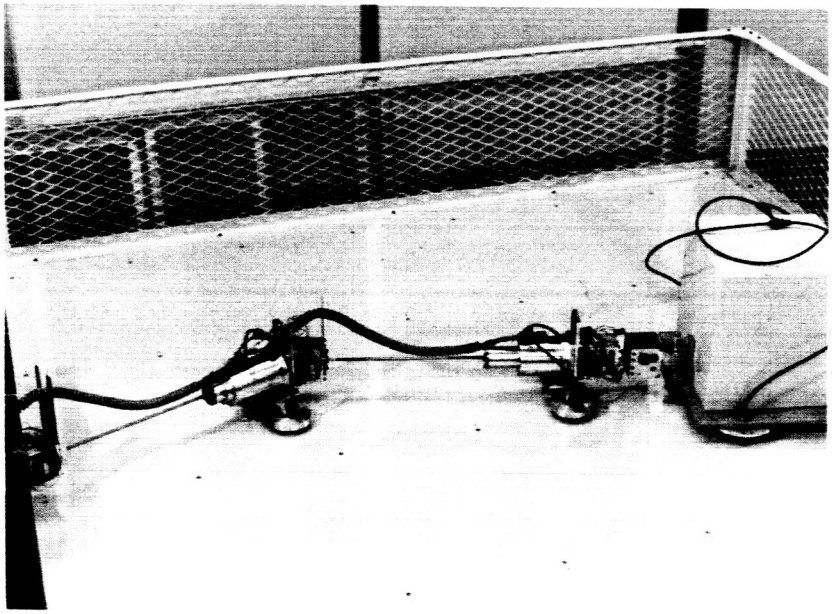


Fig. 4 TESRA-I Configuration

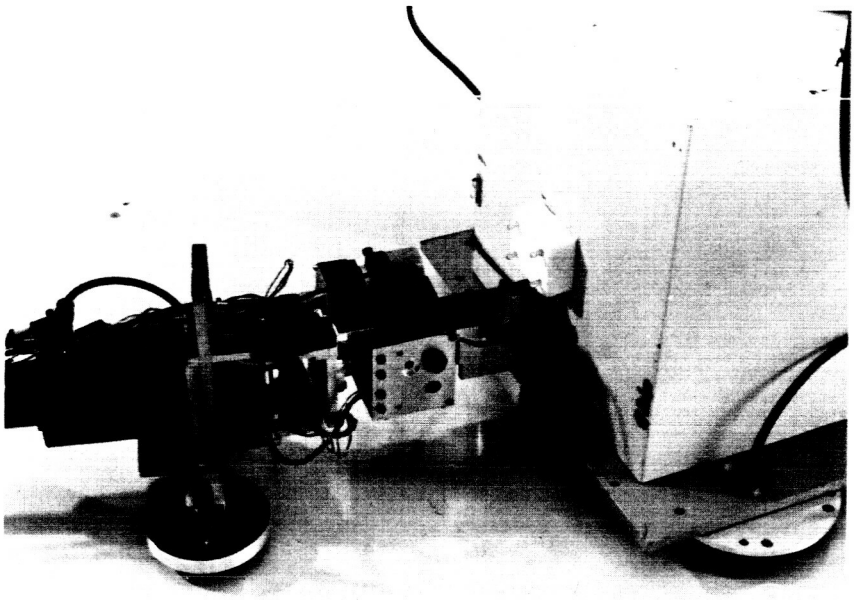
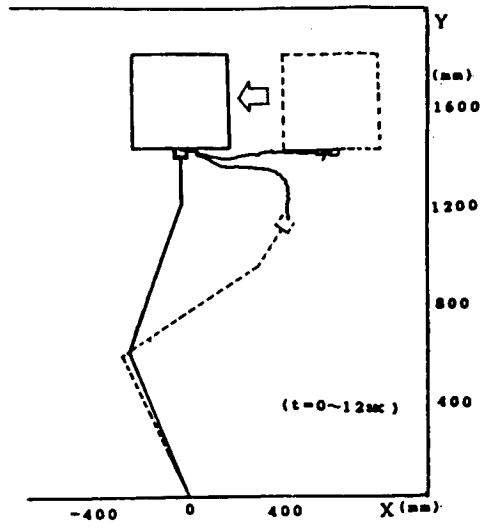
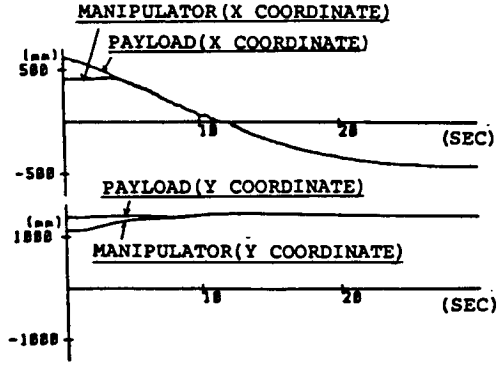


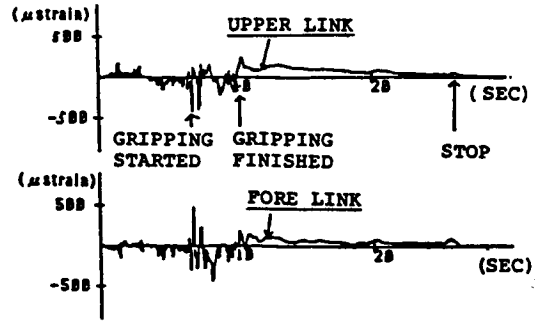
Fig. 5 CCD Camera and Target Marker



(a) Tracking Path in the XY Plane



(b) Position of a Manipulator and a Payload



(c) Link Vibration

Fig. 6 Experimental Results

TECHNOLOGY AND TASK PARAMETERS RELATING TO THE EFFECTIVENESS OF THE BRACING STRATEGY

Wayne J. Book, J. J. Wang

The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

1. Introduction

The bracing strategy has been proposed in various forms [1] as a way to improve robot performance. One version of the strategy employs independent stages of motion. The first stage, refered to here as the large or bracing arm, carries the second stage of motion. After the first stage has completed its motion it is braced to provide a more rigid base of motion with a more accurate relationship to the parts to be manipulated. The hypothesis of this research is that more rapid completion of certain tasks is possible with lighter arms using the bracing strategy. While it is easy to make conceptual arguments why this should be so, it is less easy to specify even approximately when this will be true for some reasonably generic situation. There is no relevant experience base with bracing arms to be compared to non-bracing arms. Furthermore, if one were interested in obtaining such practical, or at least relevant, experience, there would be no methodical guidance on the selection of an interesting case.

An "interesting case" is one in which the unproven approach, bracing in this paper, can show its superiority. If one such case exists, only the extent of applicability of the new approach is in question. One set of "interesting cases" is likely to be applications in which a large workspace must be covered, but where a series of small accurate moves will remain within a smaller region of the total workspace. A prototype application with these characteristics will be set up and a skeleton design of arms using the competing strategies will be compared.

2. The Problem Studied

This paper compares two operational and design strategies to pick and place a stack of n parts as depicted in Fig. 1. The first strategy employs a single arm l_1 which moves through an initial distance d_m to the part location. It then repeatedly moves a distance d_p to relocate the n parts at the final location. The second strategy employs two arms. The first arm of length l_2 carries the second arm to a bracing position from which a second short arm of length l_s to a bracing position from which the short arm can complete the n part relocation moves. The question to be answered is: Under what combinations of task parameters, technology capabilities and performance measures should one consider the bracing strategy.

The most relevant task parameters seem to be the distance of the initial move, D_m, the distance of the repeated moves, D_p the number of repeated moves n, the payload mass mp, and the size of the workspace to be covered, represented by a length of the single arm, l_1. The task chosen is also representative of other tasks in which operation is concentrated for a time within a small region of the total workspace.

The performance measures examined are the time to complete the task and the weight, under certain constraints to be described later.

The level of technology employed affects the study in several ways. The elastic modulus, maximum stress, and material density are three parameters of the material technology, for instance. These are held constant in the results presented here at values found in common engineering materials, in this case aluminum. The structural technology combines the material parameters and the task parameters to determine structural natural frequencies, mass moments of inertia, and stress. The structural technology represented here is a Bernoulli-Euler beam with a

uniform hollow circular cross section. The control technology is represented by the bandwidth of the fine motion control as a fraction of the lowest structural frequency. While advanced control schemes can now achieve higher fractions, this study assumes fine motion bandwidth of 1/2 of the first structural frequency when the joints are clamped as is representative of the limits of standard joint PID control of actuator torque. Gross motion is assumed to be limited only by the actuators and the load inertia, both of which are based on structural strength. The actuator technology represented here is the moving coil d.c. motor with a gear reduction. The motor is chosen for minimal weight to meet the peak power demands and the reducer is chosen based on required output torque.

Complete, multi-degree of freedom arm designs would be convincing to the reader in making this comparative study. However, it would be extremely time consuming and would involve an immense number of irrelevant and distracting decisions. The current study involves a far simpler design intended to capture the essence of the realistic case. The results are qualitatively applicable to real arms and can hopefully be calibrated with existing complete designs. Each arm is represented by a single beam and a single joint. For the bracing case, the large arm and

List of Symbols

a_{max} : Maximum arm acceleration
D_a : Positioning accuracy
D_m : Distance of large move between task areas
D_p : Distance of small move for fine tasks
d : Arm diameter
d_s : Short arm outside diameter in bracing strategy
d_1 : Nonbracing single arm OD
d_2 : Bracing arm in bracing strategy
E : Young's Modulus
I : Area moment of inertia
J : Moment of inertia of arm with payload
K_p : Position feedback gain in fine motion control
K_r : Ratio of arm inner and outer diameter
l : Arm length
l_s : Short arm length
l_1 : Nonbracing arm length
l_2 : Bracing arm length
M_d : Harmonic drive weight
M_m : Motor weight
M_s : Short arm system weight
m_p : Payload mass
n : Number of parts to be moved
P_{max} : Maximum motor power required in task motion
P_{fmax} : Maximum motor power required in fine motion
P_{gmax} : Maximum motor power required in gross motion
s : Fatigue strength of arm material
T_{qmax} : Maximum torque allowed
t_f : Fine motion time
t_g : Gross motion time
t_t : Total task time
θ_a : Fine motion range of arm with payload at the end point
θ_f : Fine motion range of the task
θ_g : Gross motion range of the task
θ_t : Complete task range
u : Arm mass per unit length
V_{imax} : Arm initial velocity entering fine motion range

V_{fmax} : Maximum arm velocity in fine motion
V_{gmax} : Maximum arm velocity in gross motion
W_c : First natural frequency of clamped arm
W_s : Servo bandwidth of fine motion feedback control system

3. Fine Motion and Gross Motion of a Single Link Arm

In both the bracing and nonbracing strategies the proposed task is accomplished through a series of single link moves. Task completion time is the accumulation of time for these single link moves. As a basis for the evaluation of proposed task performance, a general single link motion is first defined and analyzed. The results will then be utilized for the performance comparison of bracing and nonbracing strategies.

General arm motion can be considered as a combination of fine motion and gross motion [2]. Fine motion is defined as that part of the complete task motion commanded by the a linear feedback control law to move the arm joint to the desired position with a certain accuracy. This fine motion therefore occurs only within a certain range, which is called fine motion range, θ_f which is determined by the actuator capacity and feedback control law. Motion outside this fine motion range is called gross motion. Gross motion thus precedes fine motion and gross motion range is the whole desired task motion range, θ_t less fine motion range. It is obvious that depending on the range of the task, gross motion might not be required. To reduce task time, it would be desirable to move through the complete gross motion region as fast as possible using the maximum available torque and then in the fine motion range let the feedback control lead the arm to its final position with the specified accuracy.

Gross motion in this study is designed so that arm is accelerated and decelerated with maximum motor torque T_{qmax}. By virtue of light weight arm design and neglecting the gravity force, T_{qmax} is determined by s, the arm material fatigue strength for infinite life cycle. For a circular arm cross section of outside diameter d, area moment of inertia I, the maximum allowed torque:

$$T_{qmax} = \frac{2Is}{d} \tag{1}$$

The fine motion feedback control law in this study is linear PD control which uses joint position and velocity feedback with final desired position as a step input. The resulting system performance will be evaluated as a second order linear system. With only the joint variables available to the controller, flexibility inherent in a light weight arm generally will cause difficulty in maintaining end point position accuracy. To provide fast arm joint response and at the same time be able to damp out and avoid exciting flexible vibration, a general design criteria [3] suggests that system servo bandwidth ω_s be chosen to be half of the first clamp-free natural frequency ω_c of the arm with payload attached. An approximation for the first natural frequency is given by Den Hartog [4]:

$$W_c = \sqrt{\frac{EI}{l^3(0.23ul+mp)}} \tag{2}$$

where E : Young's modulus
l : Arm length
u : Arm mass per unit length
m_p: Payload mass

System servo bandwidth W_s is determined by position feedback gain K_p and system total inertia J and is set to be $W_c/2$.

$$W_s = \sqrt{\frac{K_p}{J}} = \frac{W_c}{2} \tag{3}$$

Based on the fine motion definition above, fine motion range for the arm with payload is:

$$\theta_a = \frac{T_{qmax}}{K_p} \tag{4}$$

Substituting (1) and (3) into (4), θ_a can be expressed as a function of arm characteristics and payload:

$$\theta_a = \frac{8sl(0.23ul+m_p)}{Ed(1/3ul+m_p)} \tag{5}$$

When payload mass m_p is much larger than arm mass, Eq. (5) can then be reduced to:

$$\theta_a = \frac{8sl}{Ed} \tag{6}$$

Arm fine motion range is shown to be only a function of material properties s, E and arm slenderness ratio l/d.

Arm fine motion range θ_a is a characteristics of the combined arm structure and payload. Depending on the task and the arm used, the task fine motion range θ_f might be equal to or smaller than θ_a. If the task range θ_t is larger than arm fine motion range θ_a, the task fine motion range will be $\theta_f=\theta_a$ and the gross motion range will be $\theta_g=\theta_t-\theta_a$. On the other hand, if θ_t is less than or equal to θ_a, the fine motion range will be $\theta_f=\theta_t$ and gross motion will not be required. To illustrate this, a single arm task is shown in Fig 2 , where a payload of 60 lb is to be moved a linear distance of 3 ft by an arm of fixed cross section for various arm lengths. The task range here is the angle the arm has to move to accomplish the task. The results show how the task range θ_t, task fine motion range θ_f, task gross motion range θ_g and arm fine motion range θ_a vary as functions of the arm length. The gross motion range decreases rapidly with longer arm since task range gets smaller and the arm fine motion range increases.

4. Fine Motion and Gross Motion Time

With the gross motion and fine motion and their control strategies defined above, performance time for each task move can be evaluated.

To avoid overshoot and achieve fast response in fine motion, the second order linear system of fine motion is set to have critical damping. For an arm to move θ_f, $\theta_f <= \theta_a$ to within an end point accuracy of D_a with θ_f as a step input, the fine motion time t_f:

$$\ln(D_a/l) = \ln(\theta_f + (\theta_f * W_s - V_i) * t_f) - W_s * t_f \tag{7}$$

where

V_i: Arm initial velocity of fine motion. $0 <= V_i <= \theta_a * W_s$

V_i is equal to zero when gross motion is not required for the task and the arm will start the fine motion from rest. Larger fine motion initial velocity, which is also the gross motion end velocity, will certainly reduce gross and fine motion time. However, it is limited by $V_{imax}=\theta_a * W_s$ to ensure that overshoot will not occur and the fine motion control torque will remain within T_{qmax} during the transition from gross motion bang-bang control to fine motion feedback control.

Depending on the gross motion range θ_g, gross motion will fall into one of the three categories discussed below:

1): $\theta_g < (1/2) * V_{imax}^2 / a_{max}$, $a_{max}=T_{qmax}/J$

This indicates that gross motion range is too small for the arm to accelerate to the maximum allowed fine motion entry velocity. Step changes in applied torque result in two jerks (bang-bang) as shown in the velocity profile, Fig

3A and 3B at one point where gross motion starts and the other point where the transition from gross motion to fine motion occurs. Fig 3A is for the case where $\theta_g < 1/8*V_{imax}^2/a_{max}$ and acceleration continues after switching into fine motion control. Fig 3B is for the case where $1/8*V_{imax}^2/a_{max} <= \theta_g < 1/2*V_{imax}^2/a_{max}$ and fine motion starts with deceleration.
Gross motion time t_g:

$$t_g=(2*\theta_g/a_{max})^{1/2} \tag{8}$$

Maximum arm velocity V_{gmax}:

$$V_{gmax}=(2*a_{max}*\theta_g)^{1/2} \tag{9}$$

2): $\theta_g >= (1/2)*V_{imax}^2/a_{max}$

This is the case where the arm is able to reach the velocity V_{imax} within gross motion range. Deceleration is required to reduce the velocity to V_{imax} when the arm enters the fine motion range. As in previous case, two step changes in applied torque are experienced with one at the start of gross motion while the other at the point when maximum reverse torque is applied to decelerate the arm to prepare for the fine motion, Fig 3C. There is no jerk at the transition from gross motion to fine motion since the fine motion control also commands a maximum reverse torque at this point due to the choice of maximum fine motion initial velocity, $V_{imax}=\theta_f*W_s$.
Gross motion time t_g:

$$t_g=(4*\theta_g/a_{max}+2*V_{imax}^2/a_{max}^2)^{1/2}-V_{imax}/a_{max} \tag{10}$$

Maximum arm velocity V_{gmax}:

$$V_{gmax}=(t_g+V_{imax}/a_{max})*a_{max}/2 \tag{11}$$

3): $\theta_g=0$

Gross motion is not required when task range is smaller than the arm fine motion range. Fine motion will have zero initial velocity, Fig 3D. One jerk exists at the start of the motion.

The total time T_t for each task move is the sum of T_f and T_g:

$$t_t=t_g+t_f \tag{12}$$

5. System Weight

System weight is the total of arm, motor and reduction gear drive weight. It is also considered as a measure of performance comparison for different arm designs. Motor technology is represented here by DC moving coil motor and a harmonic drive is chosen as reducing gear component. Neglecting motor and harmonic drive inertia, an equation to calculate DC moving coil motor weight M_m is approximated by Sangveraphunsiri [5]:

$$M_m=P_{max}/50 \text{ lb} \tag{13}$$

P_{max} is the larger of maximum power required for gross motion, P_{gmax} and fine motion P_{fmax}.

$$P_{gmax}=T_{qmax}*V_{gmax} \tag{14}$$

$$P_{fmax}=T_{qmax}*(\theta_f/\theta_a)*V_{fmax} \tag{15}$$

where $T_{qmax}*(\theta_f/\theta_a)$ is the maximum torque commanded in fine motion by feedback control.

By fitting typical harmonic drive data, its weight M_d as a function of torque can be calculated according to:

$$M_d = \frac{*T_{qmax}*\theta_f/\theta_a)^{1.027}}{467.735}$$

6. Single Arm Performance Analysis

The above quantitative gross and fine motion characterization of arm motion will serve as the basis for the following analysis of task performed by arms of different design.

The same task described earlier will be used again here. Referring to the schematic in Fig 2, an arm of circular cross section with the ratio of inner to outer diameters, $K_r=0.9$ is required to move 3 ft a payload of 60 lb.

Fig 4 and 5 show how the task completion time and system weight change as function of arm length for several arm OD's. A break down of task time is shown in Fig 6 for the case of OD=2 in. Notice the correspondence between this figure and Fig 2 showing the gross/fine motion range of the same structure and task. It is clear from these figures that a shorter arm of fixed OD has better time performance but greater total weight since greater speed and torque are used. The time performance of a more rigid arm (larger OD) is better and less affected by arm length than that of a lighter arm, however, at the cost of greater system weight, especially for the short length arms.

Another way to analyze system performance is to impose the task time requirement for each arm of various length and see how the arm OD, motor and gear reducing components will vary to meet the different time requirements. For the same previous task, the results are shown in Fig 7 and Fig 8 for the task time of 1, 2 and 3 seconds. A breakdown of system weight is presented for the task time of 2 sec in Fig 9. For any specified task time the shorter length arm requires lower system weight. Although this advantage is not quite obvious for t=2, or 3 sec, where lighter arms are used, it gets quite significant as task time is further reduced to 1 sec or less.

It can be concluded from the above analysis that a shorter arm is more effective in both time performance and total weight. Although detailed examination shows that the most effective arm length is not the shortest that can reach both points but almost the shortest, it is still fairly accurate to say that for a given task the best arm has a length of half task distance D_p with an OD dictated by the specified task time.

It would be useful to know how task time changes with variations of OD given the best arm length, $l=D_p/2$ so that task time can be reasonably specified. Fig 10 shows the task time as a function of arm OD with task distance D_p as a parameter and $l=D_p/2$. For small D_p (D_p=2 ft, l=1 ft), ODs of 1 to 2 in. yield about the same time performance as heavier arms but with much lower system weight required as shown in Fig 11. As D_p gets large (D_p=16 ft, l=8 ft), ODs of 1 to 2 in. become too flexible and result in poor time performance while larger OD (OD=3 in.) provides significant time improvement without much penalty in system weight. An arm designer therefore will have to carefully evaluate the trade-off between arm OD, system weight and task time specification.

The effects of positioning accuracy, D_a and payload mass, m_p are shown in Fig 12. Arms of shorter length and larger OD are less affected by payload variation and accuracy requirement.

7. Bracing Strategy in Large Work Space

As the study of single arm performance suggests, an arm with a length of half the task distance is the most effective. However, there are cases where it is neither practical nor possible to station a robot arm at the desired location. One of such cases, as mentioned in the beginning of this paper, is when the major task areas of D_p are within a large work space and separated from each other a distance of D_m (Fig 2). The major fine task here is doing n moves of distance D_p with payload of m_p as described before. Using a single arm which is long enough to move between the task areas and perform the fine task within each task area is certainly not an efficient design for the fine task. Bracing strategy suggests that the optimum arm design can still be applied to achieve the best fine task performance if a bracing arm is provided to move the short arm to its desired location and brace it to a rigid surface to do the fine task. Given equal system weight, the short arm will certainly outperform the single

nonbracing arm as found in the single arm analysis. However, bracing strategy carries some penalties. For the purpose of this study, penalties are mainly that bracing takes extra time and bracing arm and its associated motor and drive increase overall system weight with the fact that they are only utilized once for every n fine task moves.

Techniques similar to those used for the single arm performance analysis will also be used here to compare the performances of bracing and nonbracing strategies. Task time for nonbracing strategy is the total time of one large move of D_m and the sequential n small moves of D_p performed by a single arm of length l_1. For bracing strategy, task time is the sum of the time for 1) one large move of D_m by bracing arm of length l_2 with weight of short arm system M_s as its payload, 2) the bracing action, which is assumed to complete in one servo cycle of bracing arm system with payload M_s, and 3) n small moves of D_p by short arm of length l_s. The total system weight for each arm of l_1 , l_2 and l_s is calculated as in single arm analysis.

The first comparison will take l_1=15 ft, m_p=60 lb, D_p=3 ft with a bracing arm outer diameter d_2=3 in. Bracing arm length l_2 will depend on l_s according to the configuration shown in Fig 1. Fig 13 shows how the bracing and nonbracing task time changes as a function of l_s with short arm OD, d_s and n as parameters assuming both systems being equal weight for each l_s. Although a short arm of d_s=3 in. combined with bracing arm, system A performs fine task better than that of system B with d_s=2 in., system A requires much greater short arm system as shown in Fig 14 and thus its bracing arm has poor bracing motion and action performance. Therefore, bracing strategy with system A will not perform as well as with system B for small n, especially in the smaller d_s range. Fig 15. Larger n will make system A more effective as its fine task capability is more utilized, Fig 16. A plot of ratio of nonbracing and bracing task time is shown in Fig 17. As n gets larger, bracing becomes more advantageous for both system A and B and the optimal l_s moves toward left indicating a shorter short arm is more effective for larger n. In the extreme case where n is quite large and bracing becomes insignificant, the result should agree with that of single arm analysis which has that the optimal l_s is equal to half the task distance D_p.

The next comparison uses the same task and the same arm structure with d_s=2 in. and d_2=3 in. but with a smaller work space, l_1=10 ft. Its performance, task time ratio shown in Fig 18 is compared with that of l_1=15 ft. Bracing strategy is seen not as effective as in a large work space since the single arm in the nonbracing strategy can have shorter length and therefore will perform better.

Let the short arm for the fine task be chosen having l_s=1/2*D_p=1.5 ft and d_s=2 in.. Fig 19 shows that there exists an optimum bracing arm OD, d_2 for each n. As n gets larger, optimum d_2 becomes smaller. This is due to the fact that bracing arm is not often utilized so that it does not have to be as rigid as optimum bracing arm for smaller n.

8. Conclusion

Gross motion range and fine motion range are defined and equations for task time are given to evaluate a single arm task performance. They are derived under the control strategy that bang-bang control is used for the gross motion, and the fine motion is approximated by a critically damped second order linear system with PD joint variable feedback control. For a simple pick-and-place task, an arm is found to be most effective when its length is half the task distance and its diameter can be determined based on task time requirement or system weight constraint.

Bracing strategy is analyzed as sequential single arm tasks, bracing arm motion and short arm task motion, with techniques similar to those used in single arm performance analysis. Bracing strategy is advantageous over nonbracing strategy under some combinations of task parameters and bracing arm and short arm characteristics, especially in large work space and for large number of part moves, n. n is shown to have significant effect on the optimal choice of short arm and bracing arm structures for bracing strategy.

9. Suggestions for Further Research

The following are areas which can be probed based on the results of this study to gain further understanding of effectiveness and practicality of bracing strategy for any type of task in any environment.

1. Cost measures: Including not only time performance and system weight but also cost of design, manufacturing and maintenance of system for bracing strategy.

2. Gravitational effects: examining end point deflection, reduced arm strength due to gravity force and their effects on bracing strategy.

3. Optimization of short arm length for tasks which are evenly distributed within the large work space.

4. Examining tasks which require path control such as painting and welding to see how different types of tasks will affect arm structure requirement and the bracing strategy.

This work was partially supported through the Computer Integrated Manufacturing Systems program at Georgia Tech.

10. REFERENCES

1. Book, W. J., S. Le, and V. Sangveraphunsiri, "Bracing Strategy for Robot Operation," Proc. of Ro Man Sy '84 The Fifth CISM-IFToMM Symp. on Theory and Practice of Robots and Manipulators, Udine, Italy, June, 1984, pp. 179-185.

2. Book, W.J., "Characterization of Strength and Stiffness Constraints on Manipulator Control", Theory and Practice of Robots and Manipulators, A. Morecki and K. Kedzior, eds., Elsevier North-Holland, Inc, 1977, pp. 37-45.

3. Centikunt, S., Book, W.J.,"Performance of Light Weight Manipulators Under Joint Variable Feedback Control: Analytical Study of Limitations", Proc. of the 1988 American Control Conference, June 15-17, 1988, Atlanta, Georgia, pp. 1021-1028.

4. Den Hartog, J.P., Advanced Strength of Materials, McGraw Hill, 1952.

5. Sangveraphunsiri, V., W. Book and S. Dickerson, "Considerations in Selection of DC Motors for Lightweight Arms," Proc. 1984 ASME Computers in Engineering Conference, Las Vegas, NV, June 1984.

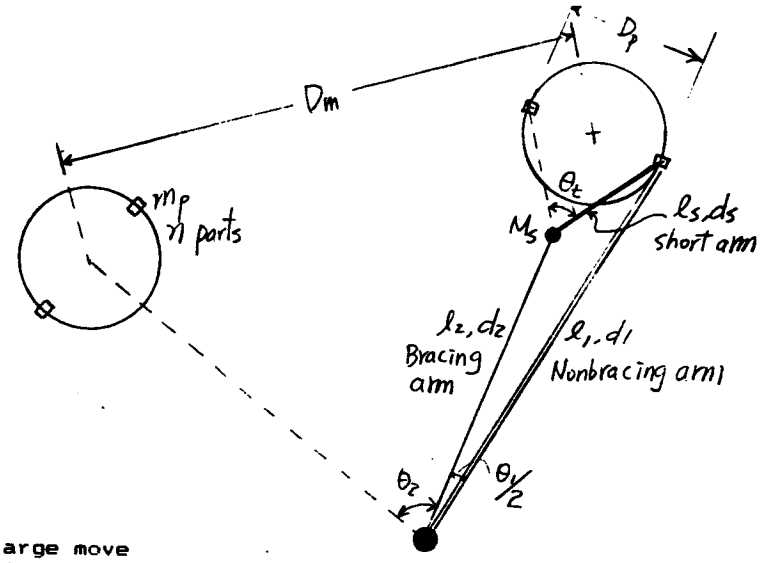


Fig. 1. Problem Task.

Task parameters:
- D_m: Distance of large move
- D_p: Distance of fine task move
- D_a: Positioning accuracy
- m_p: Payload mass
- n : Number of parts to be moved
- T : Time specification

Technology parameters:
- Material: (Aluminum)
  - E: Young's modulus
  - s: Fatigue strength s=0.002*E
  - ρ: Density
- DC moving coil motor:
  - M_m: motor weight, M_m=10.93*HP
- Harmonic drive:
  - M_d: Harmonic drive weight, $M_d=(T_{gmax}*\theta_r/\theta_a)^{1.027}/467.735$
- Control:
  - Second order linear PD joint feedback control with ζ=1, W_n=0.5W_c

Performance measures:
- Task completion time: T_c
- System weight: M

Arm structures:
- l_s: Short arm length
- d_s: Short arm OD
- l_z: Bracing arm length
- d_z: Bracing arm OD
- l_1: Nonbracing arm length
- d_1: Nonbracing arm OD
- K_r: ID/OD=0.9
- M_s: Short arm system weight
- M_b: Bracing system weight
- M_n: Nonbracing system weight

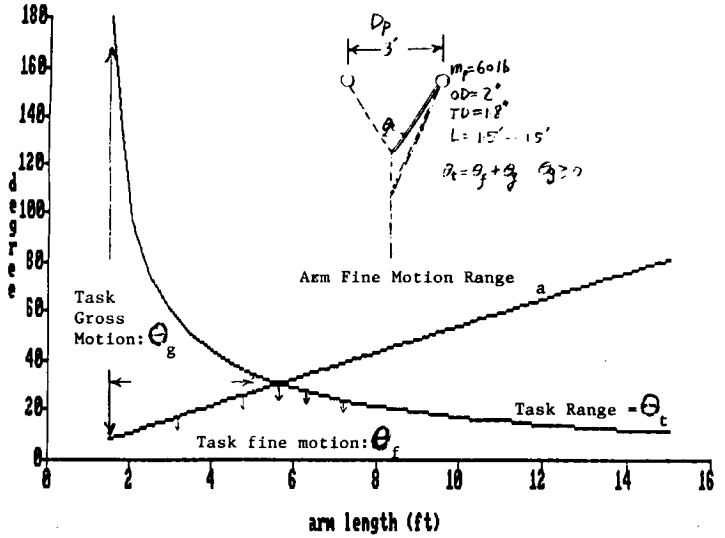


Fig. 2. Example task for a single arm: gross and fine motion range.

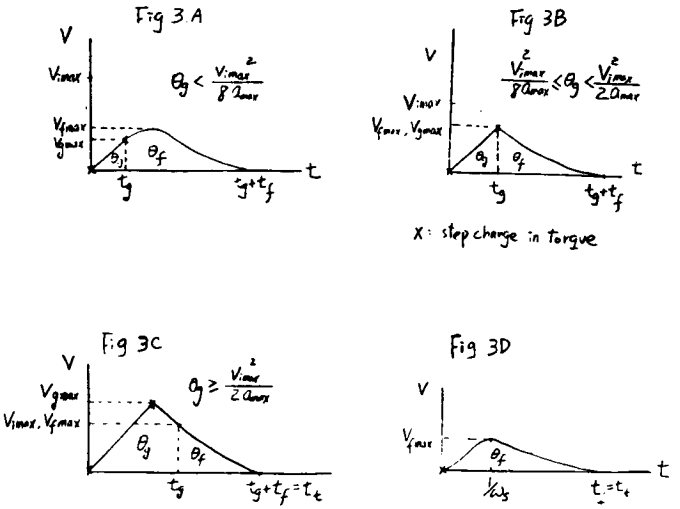


Fig. 3. Fine motion switching conditions for three cases.

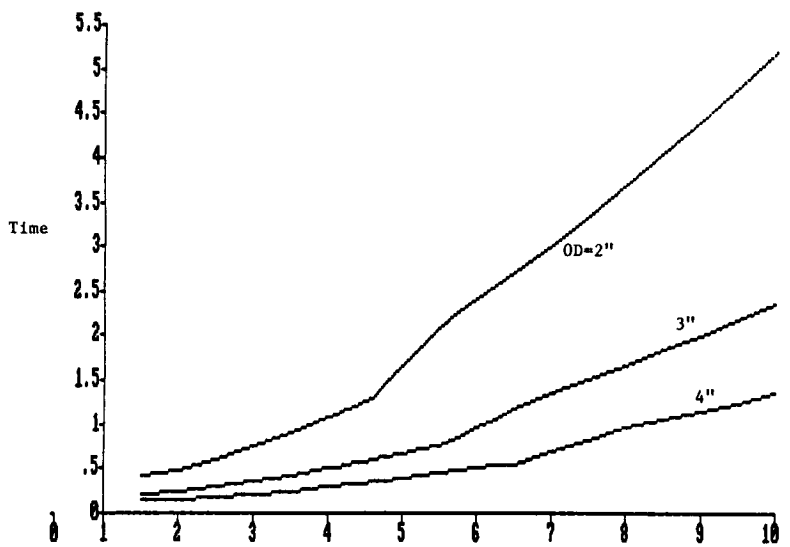


Fig. 4. Total task time for the one link case of Fig. 2.

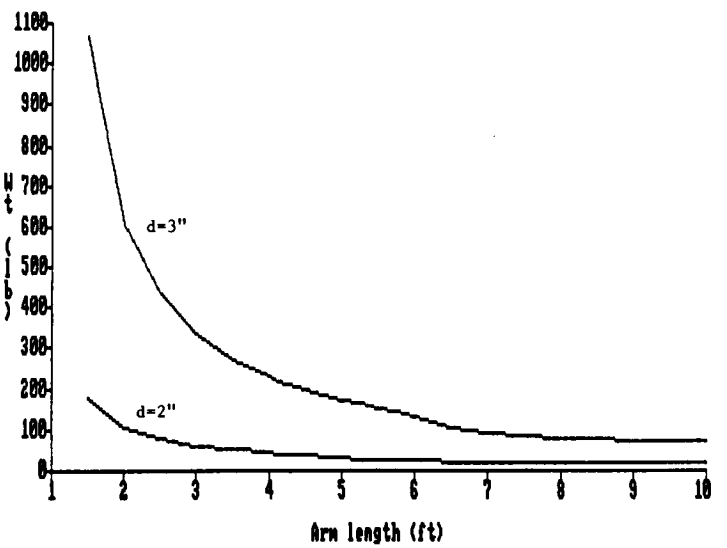


Fig. 5. Total weight for the one link case of Fig. 2.

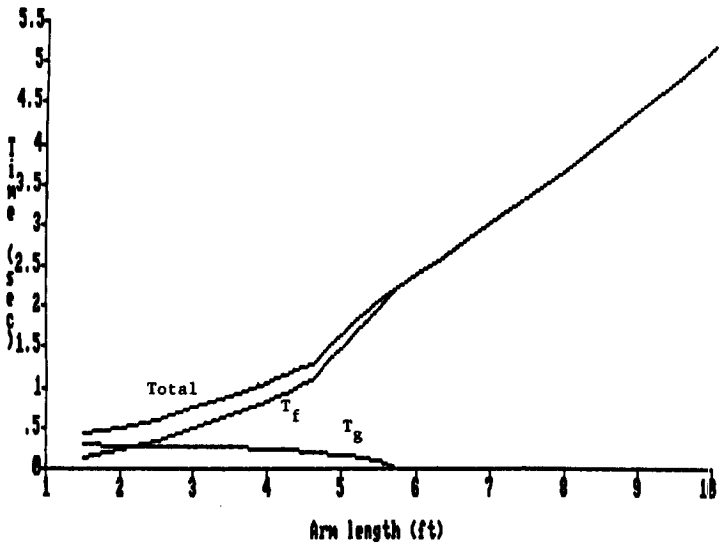


Fig. 6. Task time components for the case of Fig. 2.

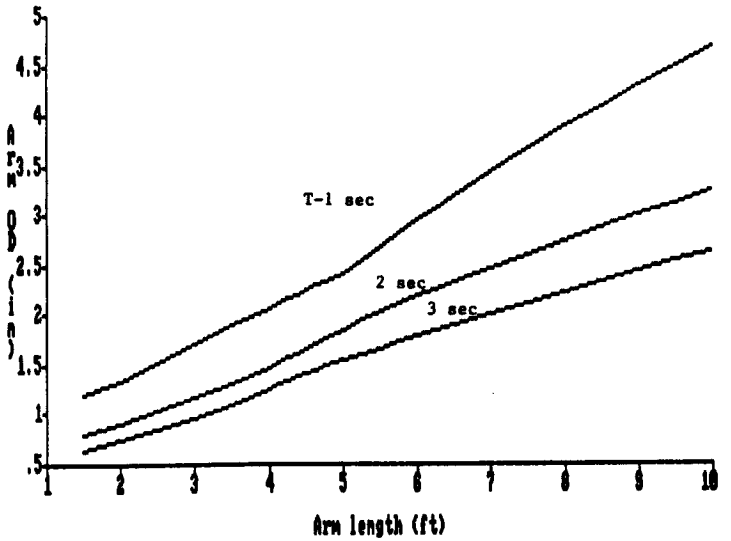


Fig. 7. Fixed task time T imposed on the one line case of Fig. 2.

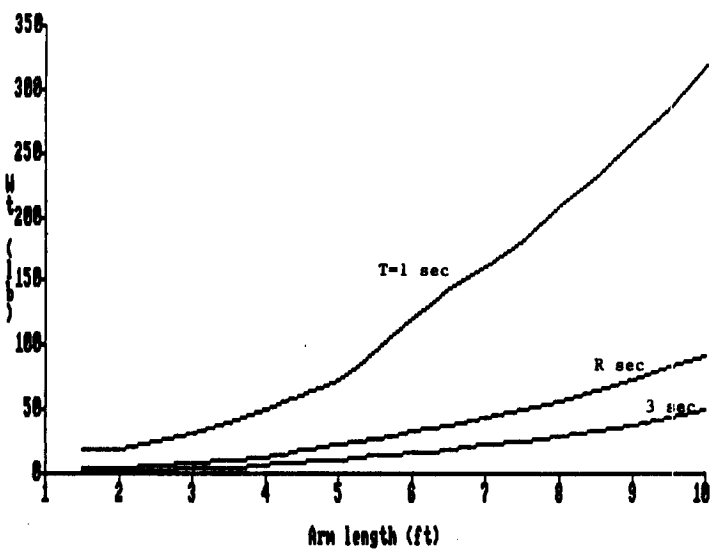


Fig. 8. Total system weight for fixed task time for the one link case of Fig. 2.

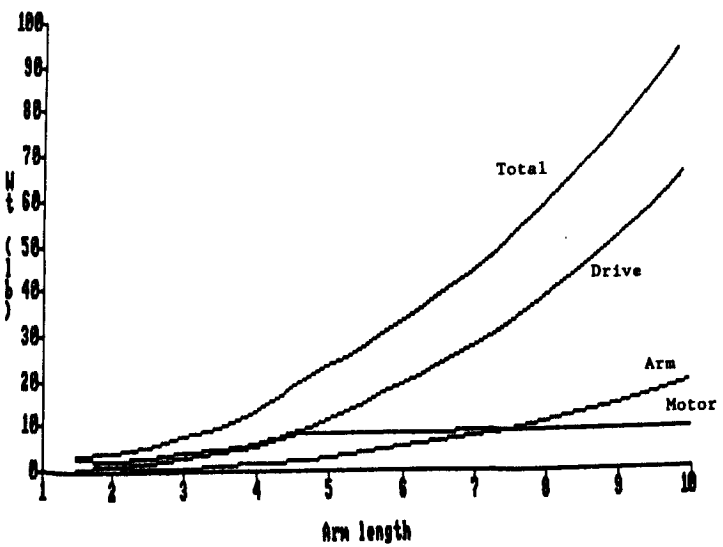


Fig. 9. System weight by component for fixed task time of 2 sec. for the one link case of Fig. 2.

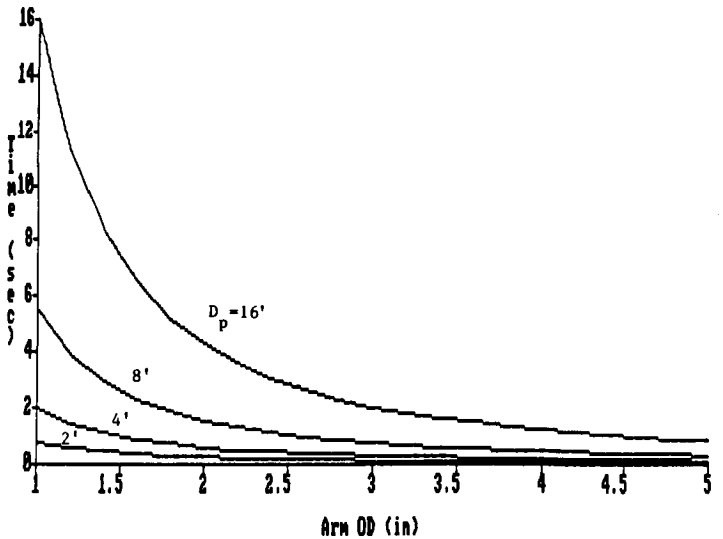


Fig. 10. Task time for fixed short arm length = $D_p/2$. (Task of Fig. 2)

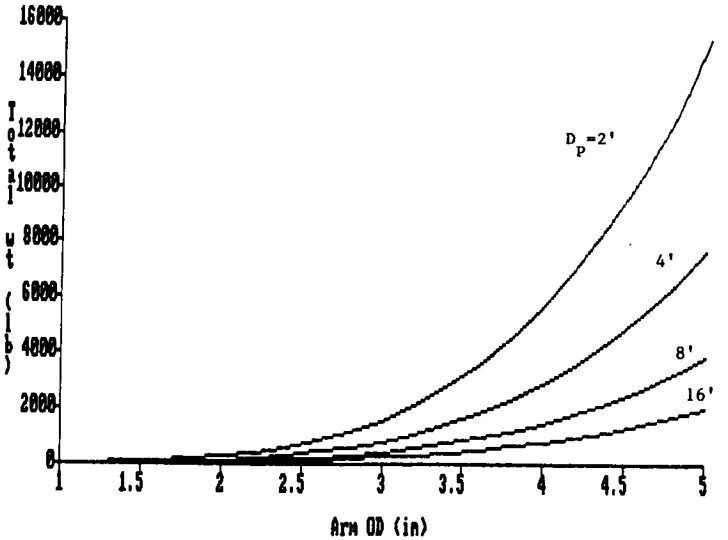


Fig. 11. Total system weight corresponding to Fig. 10.

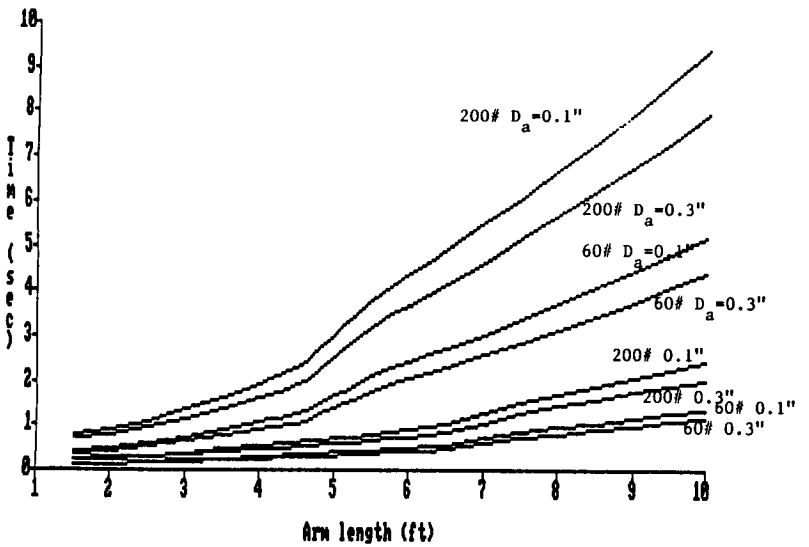


Fig. 12. Sensitivity of task time to task parameters: payload M_p and accuracy D_a. (task of Fig. 2.)

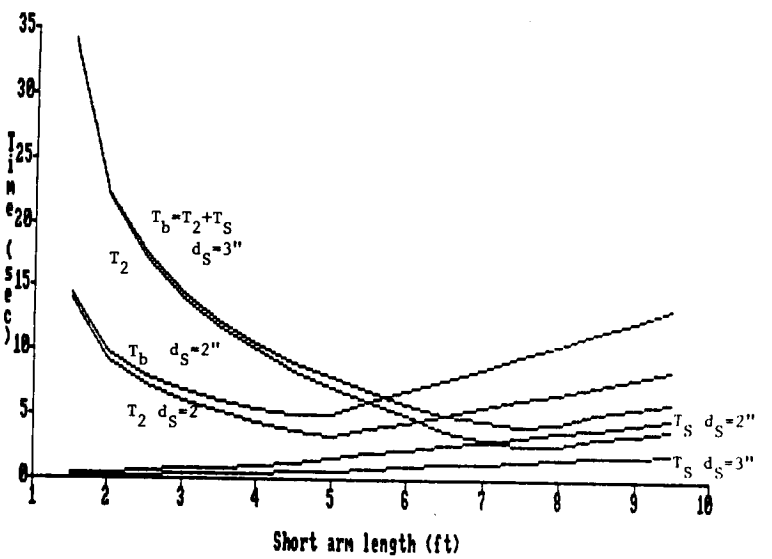


Fig. 13. Task time components T_2 and T_S for the bracing strategy for two short arm diameters, d_S. l_1 = 15', D_p=3". (See Fig. 1).

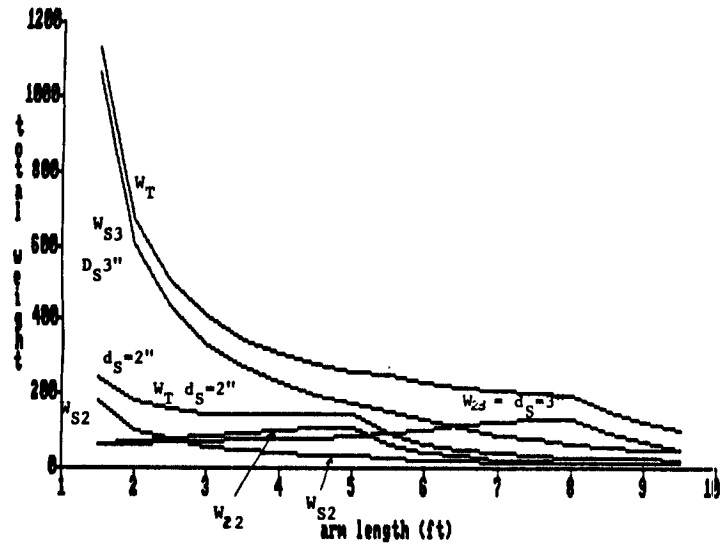


Fig. 14. Weight by component for the case of Fig. 13.

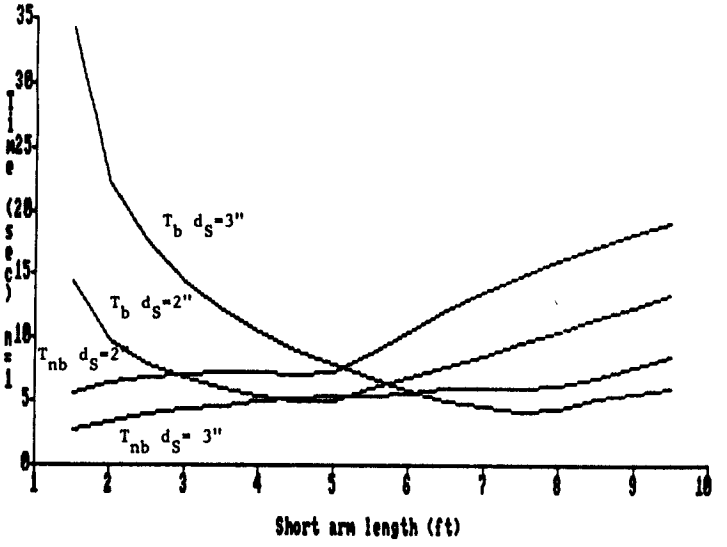


Fig. 15. Task time compared for bracing and non-bracing. (n=1)

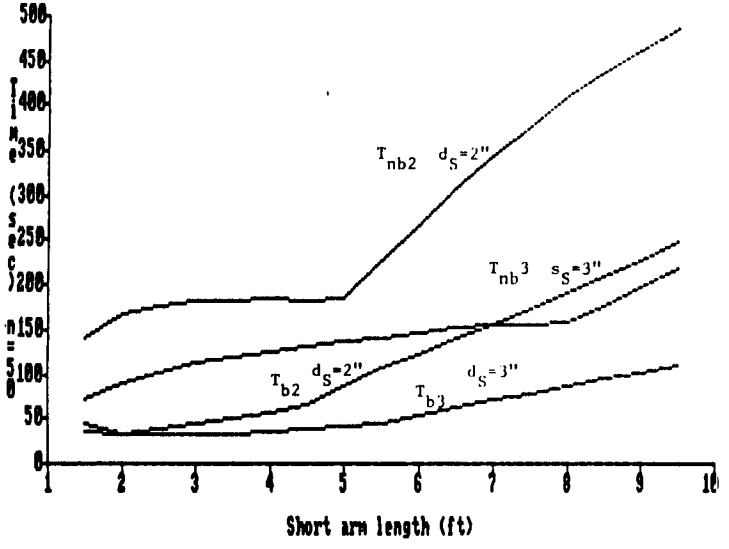


Fig. 16. Task time compared for bracing and non-bracing, (n=50).

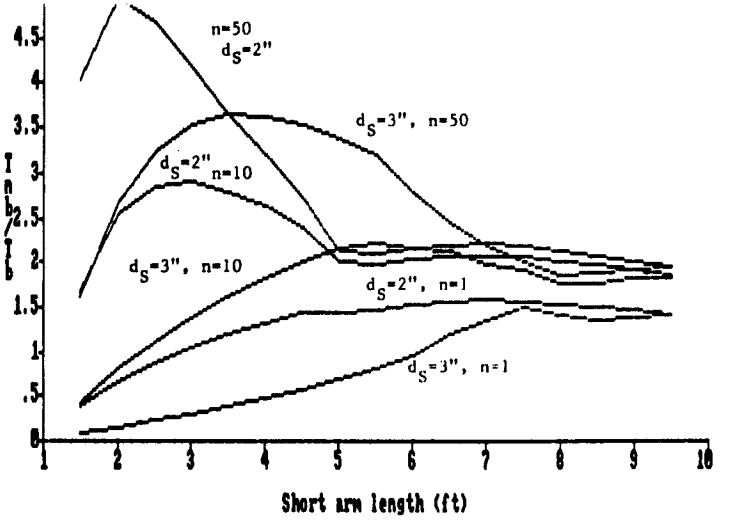


Fig. 17. Ration of non-bracing to bracing task times. (Case of Fig. 13.) l_1=15'.

Fig. 19. Optimal bracing arm OD for a fixed short arm configuration of $l_S = D_p/2 = 1.5'$, $d_S = 2''$, $l_l = 15'$, $m_p = 60$ lbs.

Fig. 18. Ratio of non-bracing to bracing task times. (Case of Fig. 13). l_l=10'.

Manipulators with Flexible Links: A Simple Model and Experiments

Isao Shimoyama
Department of Mechanical Engineering,
University of Tokyo
Tokyo, JAPAN

Irving J. Oppenheim
Departments of Architecture and Civil Engineering
Carnegie-Mellon University
Pittsburgh, PA, 15213

1. Introduction

This paper briefly reviews a simple dynamic model proposed for flexible links, and presents experimental control results for different flexible systems. A simple dynamic model is useful for rapid prototyping of manipulators and their control systems, for possible application to manipulator design decisions, and for real-time computation as might be applied in model based or feedforward control. Such a model has been proposed, with the further advantage that clear physical arguments and explanations can be associated with its simplifying features and with its resulting analytical properties.

The model is mathematically equivalent to Rayleigh's method. Taking the example of planar bending, the approach originates in its choice of two amplitude variables, typically chosen as the link end rotations referenced to the chord (or the tangent) motion of the link. This particular choice is key in establishing the advantageous features of the model: its simplicity, its efficacy, its extensibility, its physical interpretability, its observability, and its controllability. A laboratory manipulator of modular design was constructed to permit rapid link changeout and overall reconfiguration, and was used to support the series of experiments reported here.

2. Background

Multiple link manipulators are characterized by non-linear relationships between displacements in the inertial frame and displacements (rotations) in the actuator (joint) space. Inertial forces exist which are thereby non-linear and cross-coupled with joint rotations even for manipulators with perfectly rigid links. On a manipulator with flexible links *any* inertial forces have the effect of further inducing deformations (and motions) which, in their simplest form, manifest themselves as vibrations which can easily exceed in magnitude the gross intended motions of the manipulator. The configuration-dependent conditions are essentially absent in the single-link systems which have attracted much of the research attention, but our interest is in an approach which is readily extended to multi-link, three-dimensional manipulators. Moreover, multi-link manipulators are characterized by high joint masses and inertia, and a dynamic model must readily handle these concentrated mass conditions as well.

We use the phrase *dynamic model* to refer to the construct by which the *equations of motion* are to be established. Taking as an example the most simple case of an elastic prismatic link with distributed mass in

planar bending, the true equations of motion are partial differential equations which in general are not solved explicitly. Rather, an approximate solution is used to express the elastic displacement with respect to the spatial variable, yielding a set of ordinary differential equations with respect to time, which constitute the *equations of motion*. A common approach is to form the *free vibration mode shapes* and use a truncated series to produce a tractable set of *equations of motion*. However, that approach is not ideally suited to the extended set of issues cited in the preceding paragraph. The proposed simple dynamic model has features which make it well suited to these extensions, as well as being easy in application and in understanding.

3. Description of the Simple Dynamic Model

The proposed model is described elsewhere [2] and its full presentation is not repeated here. The basic physical arguments can be formulated by referring, for the purposes of discussion, to the single planar link pictured in Figure 4. The rotation θ is a tangent to the link rotation (alternately θ may denote the rotation of the *chord* between the link end-points); two further variables are denoted, ϕ and ψ, constituting the link end rotations with respect to the chord. Two physical approximations are then made:

- The kinetic energy of the link distributed mass, *m*, is approximated as that of a mass *m* translating with the center-of-mass of the link *chord* itself.
- The displacement shape, with respect to the *chord*, is approximated as the displacement accompanying static end rotations ϕ and ψ. (The differential equation governing flexure is readily solved to yield the polynomial solution for the displacement.)

This model readily represents translations and rotations of the concentrated masses as linear combinations of θ, ϕ, and ψ. Moreover, the three rotation variables are readily observable through rotation and strain sensing, and are directly coupled to the actuator inputs.

In essence, the model has introduced two *amplitude variables* (ϕ and ψ) to approximate the elastic effects. The physical nature of most vibrations is such that this choice generally models the most significant vibration effects. Moreover, the formulation is expressly compatible with assembly of equations for multi-link systems, such that the most important configuration dependencies will be modelled automatically. Its assembly and its solution (computation) are simple, and as stated above the model variables are well-matched to the control problem.

In the general case, a rigid link has its position (with respect to its local origin) expressed by three rotations. For a flexible prismatic link, the model poses the need for eight rotations; three equivalent to the rigid body rotations, two for flexural end rotations in each of the two principal directions, and one (the *relative rotation* between the end points) in torsion. In our opinion the model will be reasonably effective at approximating the *equations of motion*[1].

[1]The description applies most clearly to links which are prismatic, doubly symmetric, etc. For links which are irregular the physical reasoning used in posing the model can still apply, but the mathematics describing the displacement states must be updated; one of the experimental studies described in this paper includes such an extension, for a link of tapering cross-section.

4. Manipulator Configurations for Preliminary Experiments

A series of preliminary experiments have been performed at Carnegie-Mellon. The first was a single link in planar motion actuated by 1-DOF end rotation. The flexible link was of constant cross section with distributed mass and with concentrated mass at the tip, and the actuator was a small direct drive DC motor. The results of that experiment are described in an earlier paper [2] (with additional authors) and are not reproduced here; as expected, comparison of simulated and experimental histories confirmed a reasonable accuracy for the simple dynamic model.

The subsequent preliminary experiments in three-dimensional motion, described for the first time in this paper, were performed using a manipulator of modular design built at Carnegie-Mellon. The system features six actuators which connect through endplates and fittings to a variety of different links. This results in rapid changeout and inexpensive link fabrication; in addition to the experiments using flexible links the system has been used as a 6-DOF manipulator with rigid links operating under position control, and as a 4-DOF manipulator (using totally different configuration and link dimensions) with strain-sensing on the links operating under a supervisory level of force control. Each actuator consists of a DC motor, harmonic drive gearing, and a potentiometer for rotation sensing.

Figure 1 is a sketch of the manipulator as it was configured for the flexible link experiments. Three actuator units create a *roll, pitch, yaw* set of actuated DOF. Two flexible links were used. The first is pictured as a "fishing rod" with a tip mass; it was actuated through 2-DOF (*pitch* and *yaw*) and is further depicted in Figure 2. The second is pictured as a (flexible) "pipe" attached to a second (rigid) link; it was actuated through all 3-DOF and is further depicted in an analytical equivalent in Figure 3.

5. Results of 2-DOF Experiments

The motion of the "fishing rod" under 2-DOF actuation is a three-dimensional motion through a spherical angle, and a pilot experiment was first performed successfully by Heller [1]. The fishing rod is modelled here as a single link for which motion about the roll axis (torsional vibration) can be ignored. Figure 4 is the model of the link for motion about the pitch axis. Note that the link is of tapering cross section, and that in Figure 4 the end rotations ϕ and ψ are referenced to θ, the tangent to the link motion about its base. The inertial and friction properties of the actuators were determined by measurement and by system identification (not shown). The simple model was then applied to the link using the physical assumptions expressed earlier, and including without difficulty the variation in the cross section with length (also not shown, owing to requirements of brevity in this paper).

The basic experiment was a step motion (0.2 radians pitch rotation and 0.1 radians yaw rotation) performed under position control[2] only. Strain histories in Figure 5 evidence the resulting vibrations, and (with various other experimental obervations) show the motions to be largely independent (uncoupled) of one another. The experiments were repeated adding feedback control on the strains at the base of the link. The experiment was performed about one particular point in joint space, and gains were chosen by trial and error. Figure 6 shows the resulting rotation and strain histories, evidencing an adequate reduction of vibration.

In Figures 5 and 6 the simulated histories were generated using the results of the simple dynamic model.

---

[2]Throughout this paper *position control* or *position feedback* refers to direct feedback control of *joint rotations*.

The model appears to be reasonably accurate in predicting the system frequencies. While the model was not used to set gains in this particular case, it did establish that the joint (actuator) rotations and link base strains would constitute the required state variables for control.

6. Results of 3-DOF Experiments

The configuration for the 3-DOF experiment can be considered a two-link system (proximal link is flexible and distal link is rigid) under general three-dimensional motion which will display coupled lateral-torsional vibrations. In the first experiment the joints were clamped to behave as a rigid boundary, and the system was set into motion by being given an initial tip displacement and being released at time zero. The resulting strain histories are shown in Figure 7a. They reveal the coupling of lateral and torsional vibrations, the significant vibration amplitudes, and the minimal material damping. The system was then restored to a configuration for position and strain feedback using gains chosen by trial and error for control about that point; Figure 7b shows the effective control of all vibrations under the same experimental excitation. Figure 8a shows the rotation and strain histories under a three-dimensional 3-DOF step motion, for the case of position feedback. Significant vibrations are observed; the damping present in this case (as compared to the results in Figure 7a) results from the dynamics (and friction) of the actuators. Figure 8b shows the rotation and strain histories when strain feedback feedback (on three channels of strain taken at the base) is added, evidencing effective vibration control. Analytical studies of the 3-DOF experiments have not yet been completed.

7. Summary

The approach and physical arguments for our simple dynamic model have been discussed briefly. The proposed model has various features which are well matched to demands which surface when studying manipulators with flexible links. At this time the model is proposed for the attention, consideration, and use of researchers. A series of experiments were performed demonstrating vibration suppression using direct feedback control on end rotations and link strains. For one experiment in which analytical results have been generated, the comparison of experimental and simulated histories shows reasonable performance of the simple dynamic model in capturing system frequencies and in confirming the needed state variables.

8. References

[1] Heller, M.
Experiments in Feedback Control of a Flexible Link.
Master's thesis, Carnegie-Mellon University, 1988.

[2] Shimoyama, I., Miura, H., Komatsu, T., and Oppenheim, I.
A Simple Dynamic Model for Control of Flexible Manipulators.
International Journal of Robotics Research under review(currently), 1987.

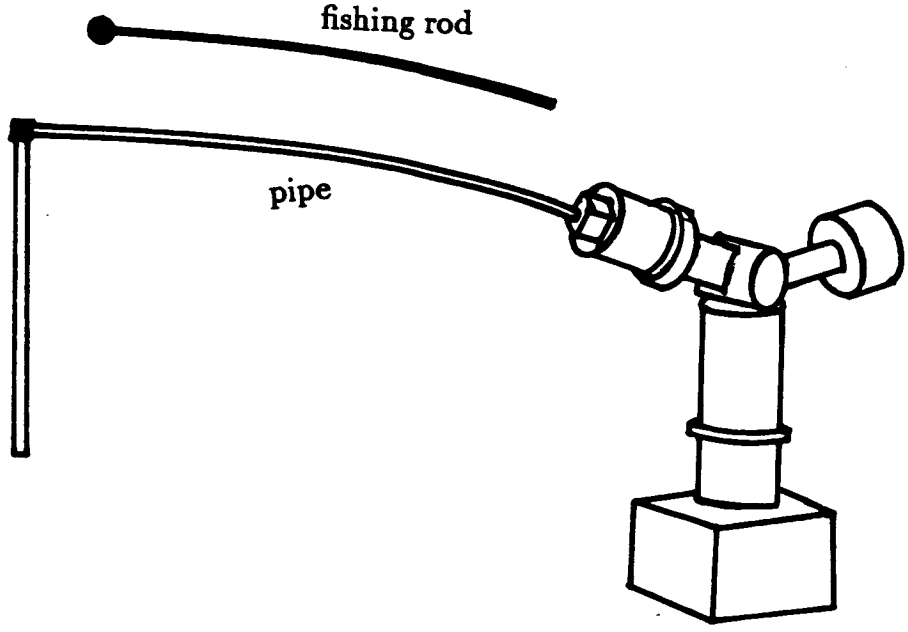


Figure 1. Manipulator as configured for 2-DOF (spherical motion, "fishing rod") and 3-DOF (coupled lateral-torsional motion, "pipe") experiment.

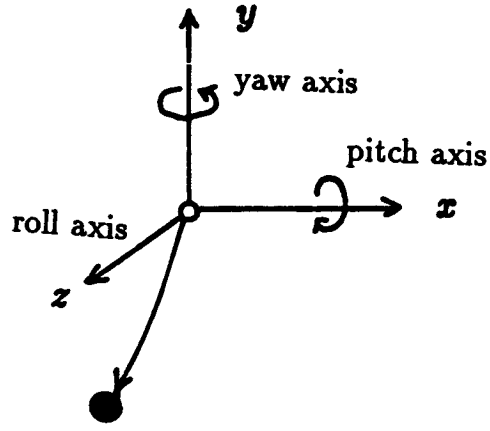


Figure 2. 2-DOF Experiment

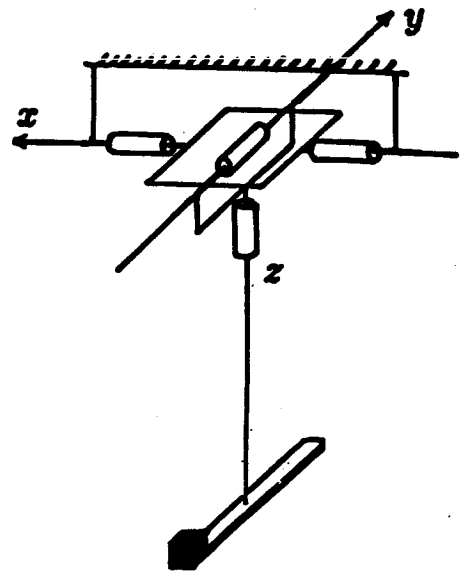
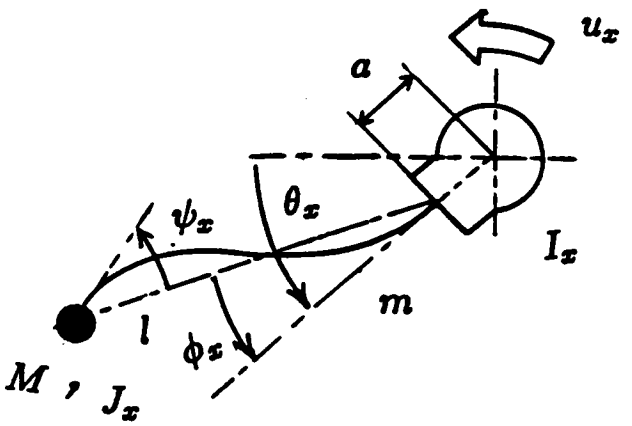


Figure 3. 3-DOF Experiment



M, m	mass
I, J	moment of inertia
l, a	length
θ, ϕ, ψ	angle
u	torque
d_0	diameter at the base end of the fishing rod
d_1	diameter at the tip end of the fishing rod

Figure 4. System Variables in the 2-DOF Experiment. (Pitch axis shown)

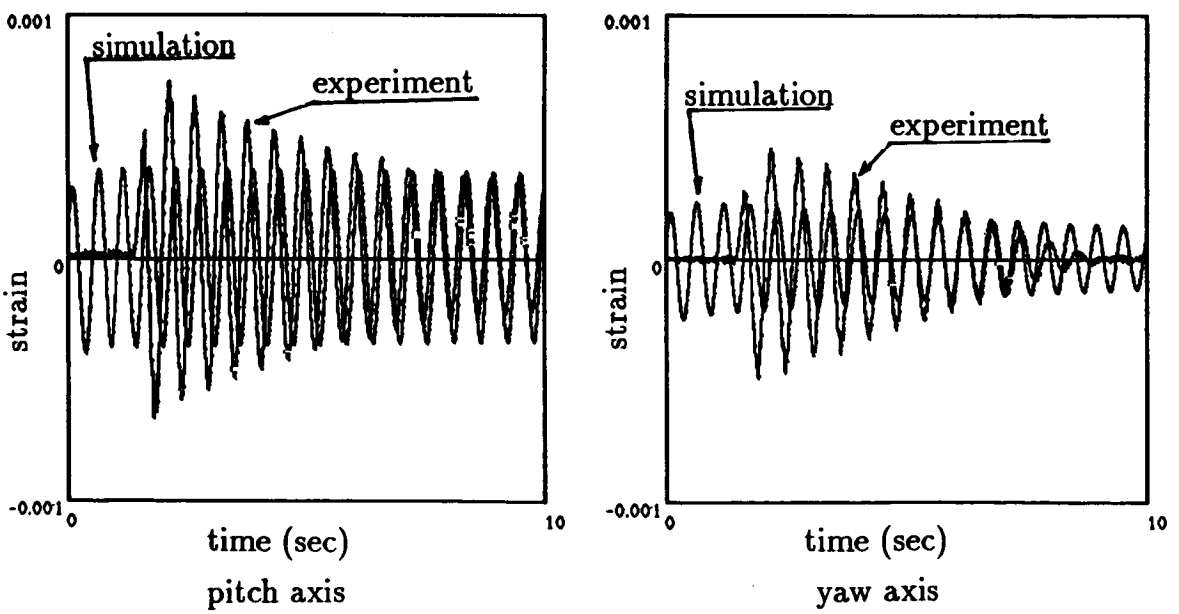


Figure 5. Strain Histories after a Step Motion, Position Feedback Only.

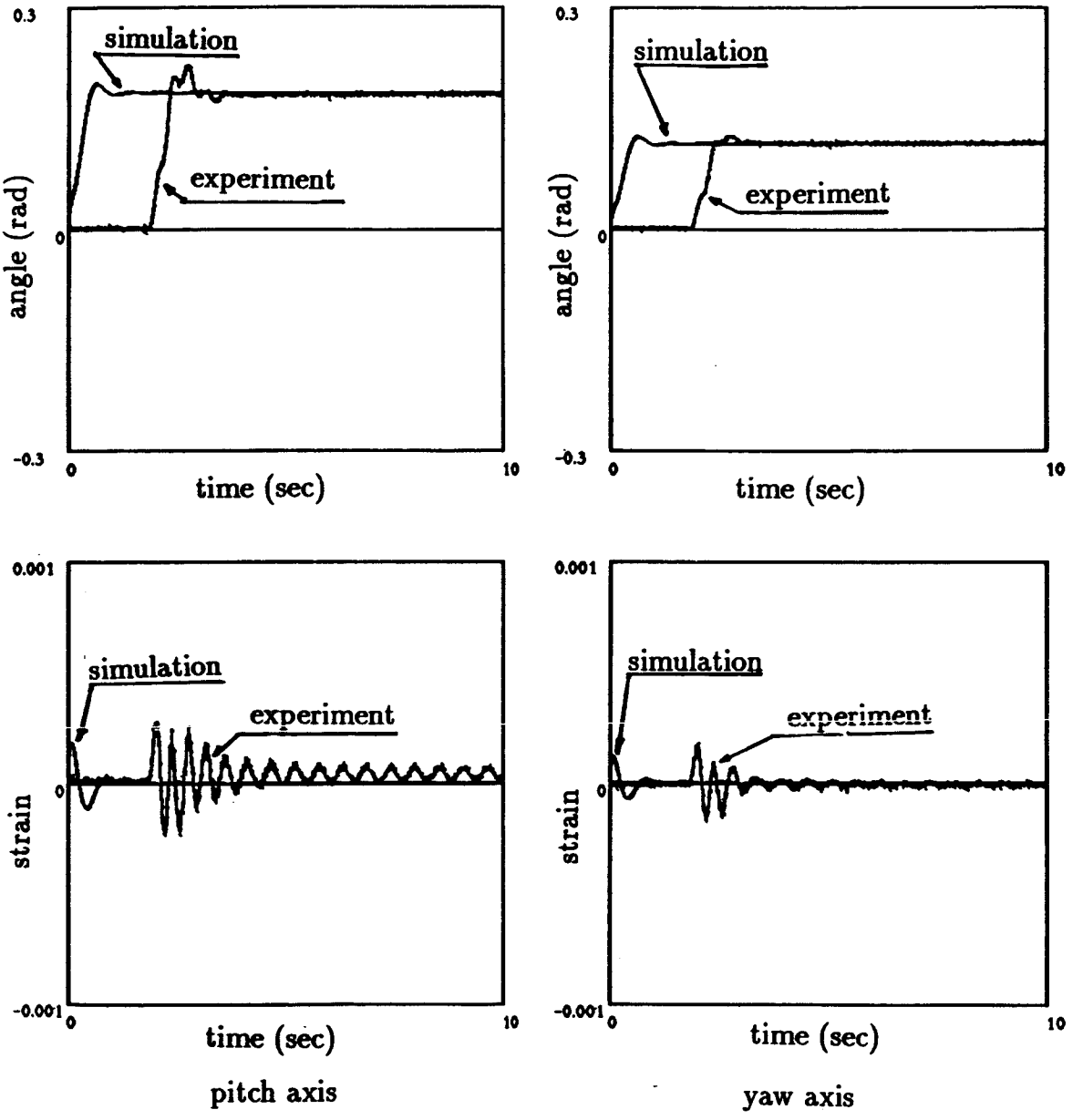


Figure 6. Rotation and Strain Histories after a Step Motion. Position and Strain Feedback.

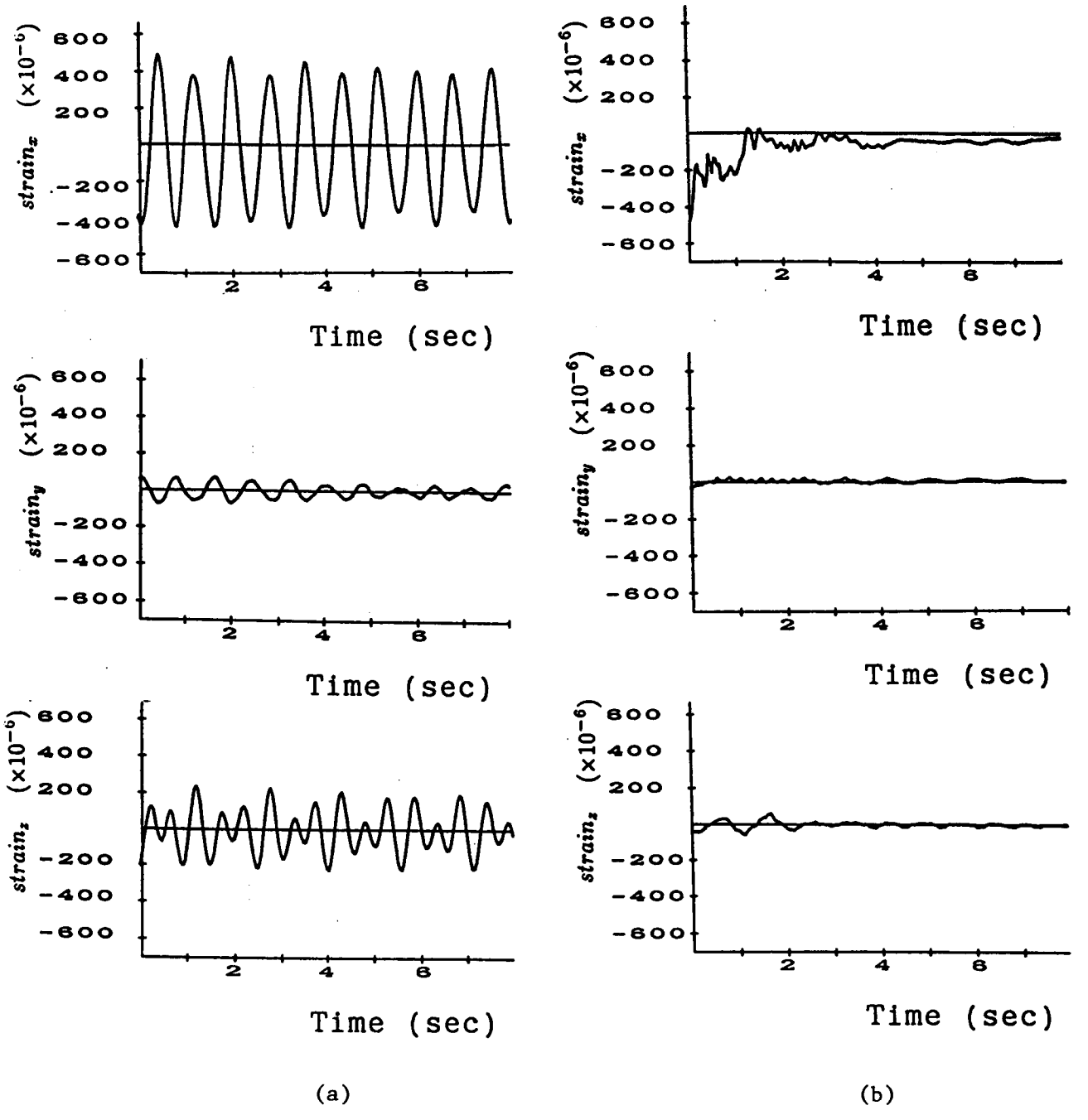


Figure 7. Strain Histories after Release from an Initial Tip Displacement

(a) Three joints rigidly fixed; coupled lateral-torsional vibrations present

(b) Three joints under position and strain feedback; vibrations eliminated

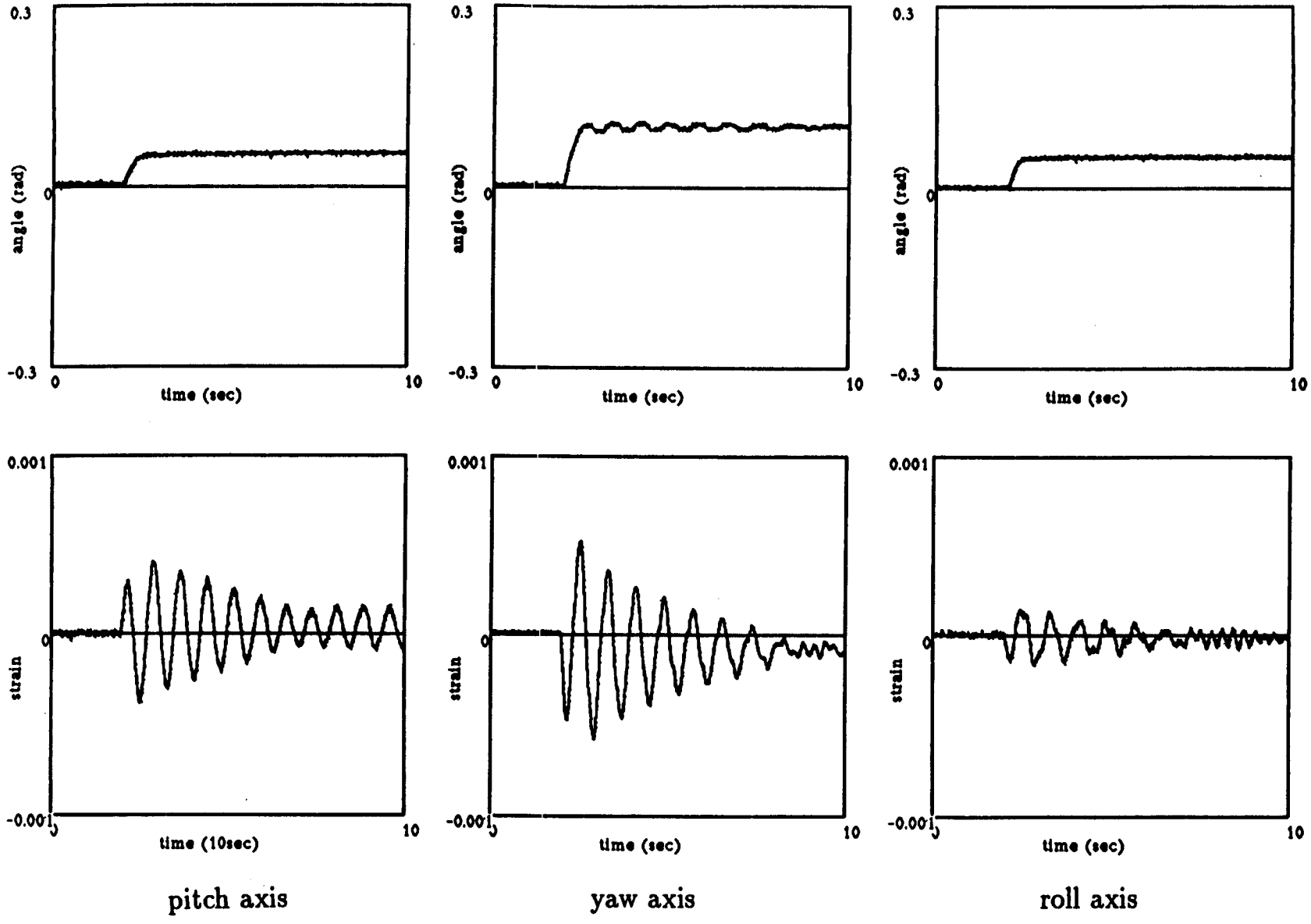


Figure 8a. Rotation and Strain Histories after a Step Motion, Position Feedback Only.

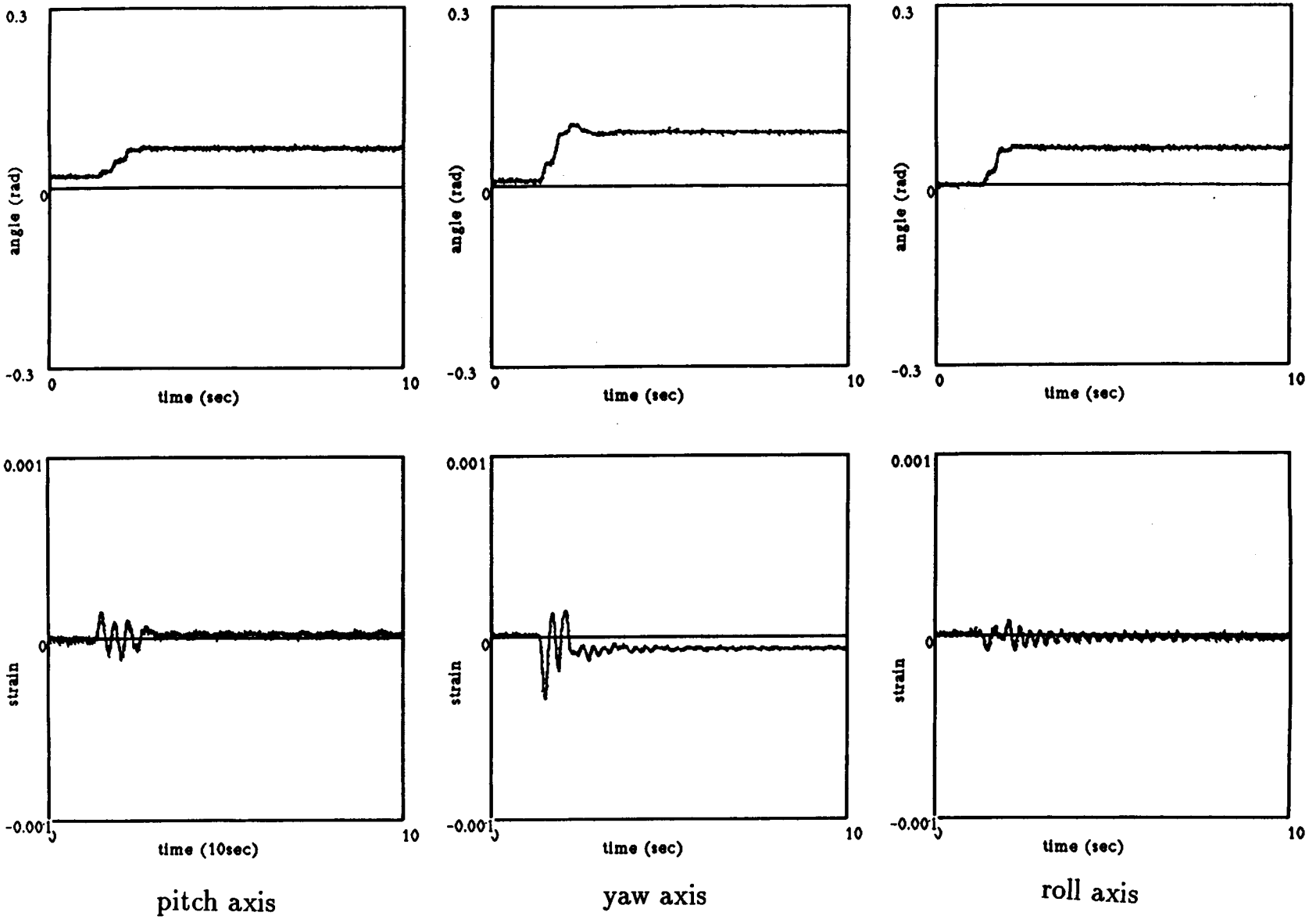


Figure 8b. Rotation and Strain Histories after a Step Motion, Position and Strain Feedback.

EXPERIMENTS IN IDENTIFICATION AND CONTROL OF FLEXIBLE-LINK MANIPULATORS[1]

Stephen Yurkovich Anthony P. Tzes Fernando E. Pacheco
The Ohio State University
Department of Electrical Engineering
2015 Neil Avenue
Columbus, Ohio 43210

Abstract

Interest in the study of flexible–link manipulators for space–based applications has risen strongly in recent years. Moreover, numerous experimental results have appeared for the various problems in the modeling, identification and control of such systems. Despite the recent activity throughout the literature for flexible–link manipulator control, relatively little has appeared involving laboratory verification of tuning controllers for realistic flexible–link manipulators which are required to maintain endpoint accuracy while manipulating loads which are possibly unknown and varying, and while undergoing disturbance effects from the environment and in the workspace. This paper reports on an ongoing effort in these areas for endpoint position control of flexible–link manipulators, with laboratory setups consisting of one and two–link manipulators.

I. Introduction

Efforts in the modeling and control of flexible–link manipulators have been motivated by the foreseen demand for lightweight, accurate, high–speed robots in space telerobotcs and several other applications. Presently, studies in these areas have reached a fairly high level of maturity, due primarily to numerous works in the last four years from both an analytical viewpoint and, to a lesser extent, experimental viewpoint. Analytical studies in modeling flexible–link robots abound, and are in fact too numerous to cite here; References [1,2] serve as excellent summaries of existing works in flexible manipulator modeling. Equally numerous are the various approaches which have appeared in the literature for controller design schemes. The greatest number of these works have dealt in simulation studies only, and some have developed quite elaborate and complex control schemes.

On the other hand, several successful laboratory setups have demonstrated the effectiveness of relatively simple algorithms for flexible manipulator control. While most experimental studies have focused on single–link manipulators, or multi–link manipulators with a single flexible link, such setups have served as valuable testbeds for modeling, system identification and controller design. Some of the more visible experimental efforts have been the work of Yurkovich *et al.* [3,4,5,6], Schmitz, Rovner, *et al.* [7,8,9], and Book *et al.* [10,11], among others. In [4] the use of measurements from a linear accelerometer in vibration compensation of the robot endpoint was shown to be extremely successful, proving the concept of acceleration feedback for flexible–link manipulator control. The

[1]Supported in part by the National Aeronautics & Space Administration under NASA Grant NAG–1–720.

use of acceleration feedback has intuitive appeal from an engineering design viewpoint, due to ease of implementation, relatively low cost, and advantages of structure–mounted sensing.

Despite this recent activity, relatively little has appeared involving laboratory verification of tuning controllers for realistic flexible–link manipulators which are required to maintain endpoint accuracy while manipulating loads which are possibly unknown and varying. This paper discusses several techniques for flexible–link systems, and presents experimental results in system identification and control of a one–link flexible manipulator carrying an unknown, varying payload.

II. Problem Description

Two laboratory setups are currently utilized in the Flexible Structures Facility at Ohio State (Department of Electrical Engineering) [12,13]. The one–link system is the subject of the experimental results reported in this paper and is described in detail below. The identification and control techniques described are, however, currently being investigated primarily for the second system which consists of two flexible links situated in the horizontal plane. The first link, which is driven by a 3.4 ft lb direct drive servomotor, is made of aluminum, 0.75 meter in length and 0.125 inches in width. Mounted at the endpoint is a 1.5 in-lb geared servomotor to actuate the smaller (0.5 meter aluminum) second link, 0.0625 inches in width. Both links are therefore very flexible, and the setup lends itself to complicated modeling, identification and control problems. A VME Bus Motorola 68020/68881 system, with 16 channels of A/D and four channels of D/A, is used as the control computer. Results of experimentation with this apparatus are forthcoming.

A. One–Link Setup

The flexible–link manipulator arm of this study is a beam made of $\frac{1}{16}$ inch 6061–T6 aluminum, one meter in length and 10 cm in height. The arm is counterbalanced about the motor axis with a rigid aluminum attachment 38 cm in length. An electromagnet device is mounted on the manipulator endpoint to facilitate experimentation with different payloads. Actuation at the hub is accomplished by a direct drive dc motor with rated stall torque at 680 oz–in. The two sensors for use in feedback control are the accelerometer, located at the arm endpoint, and the optical shaft encoder located at the hub. The encoder is rated at 3600 pulses per revolution, allowing measurement of the shaft angular position with a resolution of 0.05 degrees. The piezoelectric accelerometer is rated at $\pm 250g$ with a sensitivity of 1.15 mV/g. A linear array line scan camera is used for data recording, reading a light source at the arm endpoint, but is not used in feedback control (results of endpoint position feedback for this setup were presented in [4]). The computer used in the data acquisition and control is the DEC MicroVax II.

B. Identification and Control

The laboratory setup has furnished an excellent testbed for investigation of many ideas in the areas of identification and control, several of which are described in the sections to follow. Specifically, methodologies under study have included

- Position feedback, fixed controller designs;
- Acceleration feedback, fixed controller designs;
- Eigenstructure realization algorithms for identification;
- Auto–tuning control designs with identification, time domain;
- Auto–tuning control designs with identification, frequency domain;
- Input shaping with acceleration feedback;
- Learning schemes;

- Time optimal slewing controllers.

A characteristic of flexible–link manipulators situated in the horizontal plane is that the modal frequencies are reduced when a payload is added. Motivated by this and the fact that a fixed controller will not perform well over a range of payloads, the idea pursued in much of the techniques listed above is to tune a nominal control configuration according to the changing characteristics of the arm. As an illustration, consider the *nominal* case, that is, when no load is carried by the arm. This nominal control scheme utilizes endpoint acceleration through a static feedback gain, with shaft position in a separate static gain feedback loop. We note that more complex schemes have been investigated, such as linear quadratic regulator theory, or inclusion of dynamics in the compensation network, but the primary objective was to attain good performance with the simplest possible control technique. This acceleration feedback control scheme is very robust to disturbance effects (can maintain endpoint position even when the arm is jolted), but, as might be expected, cannot perform nearly as well for significant payload variation. This effect is verified experimentally by having the arm carry a payload weighing 0.67 lbs, which is approximately 63% of the weight of the arm itself. Figure 1 shows the results of this exercise, where the *same controller gains* utilized in the nominal, no–load case (dashed curve) are employed for the case with payload; the response is to a commanded input which basically demands that the arm endpoint follow a square wave reference. Attempts at designing fixed controllers with more complicated dynamics were only slightly more successful. Indeed, the large overshoot could be avoided, and endpoint position accuracy maintained, if the control gains were tuned appropriately. For purposes of comparison, Figure 2 offers the open loop response which illustrates the flexibility of the arm, even for a relatively small slew angle of only about 15°.

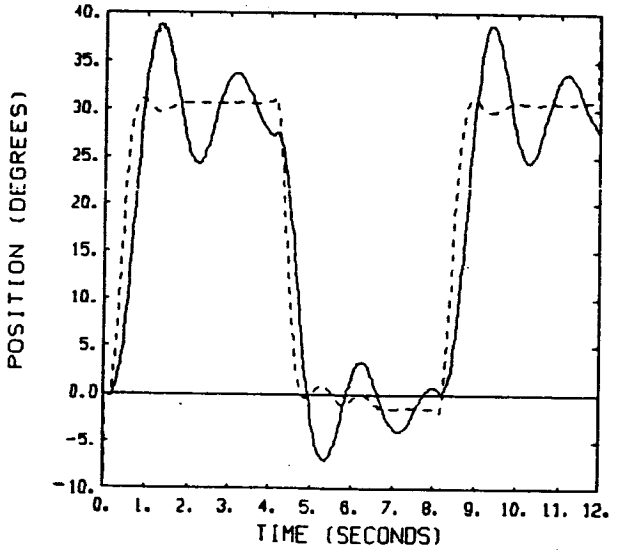


Figure 1: Effect of Payload

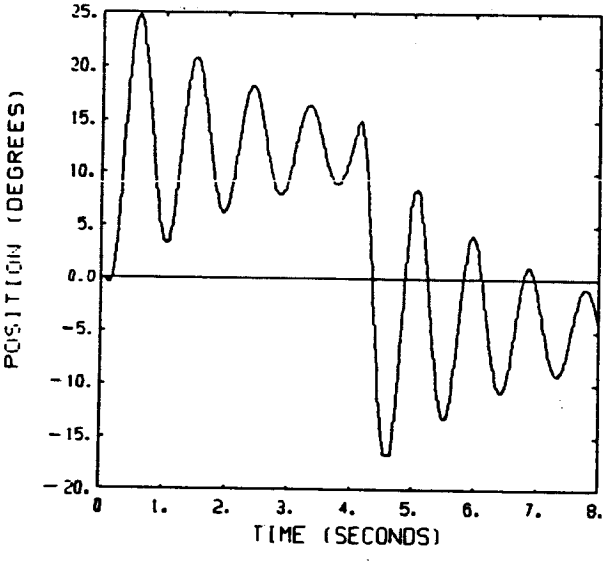


Figure 2: Open Loop (no control)

III. Time Domain Auto–Tuning Control

A. Identification

Since the control objective we consider in this problem involves vibration suppression ***after*** the manipulator has undergone the nonlinear slew maneuver, a reasonable choice of model structure

amenable to control design for the input to shaft angle and the input to endpoint acceleration transfer functions (filter) is the Autoregressive Moving Average (ARMA) model. Within this setting, consider

$$y(k) = \phi^T(k)\theta + w(k) \quad , \tag{1}$$

where θ is the vector of filter coefficients, $w(k)$ is stationary, zero mean process noise, and $\phi^T(k) = [y(k-1), \ldots, y(k-n), u(k-d-1), \ldots, u(k-d-n)]$ is the regression vector, for $y(k)$ and $u(k)$ the system output and input, respectively, and d is the inherent delay (in sampling time multiples) between the commanded input (for hub torque actuation) and the response seen in the shaft angle or, more noticeably, in the endpoint acceleration.

For filter parameter updates we limit our discussion here to the least squares and recusive least squares (RLS) algorithms. Both techniques are based on computation of the optimal value of the parameter vector θ based on minimization of a scalar loss function of the squared equation error. That is, the well-known non-recursive solution to this procedure for the model (1) is given by

$$\hat{\theta} = M\Phi^T y \quad , \tag{2}$$

where the information matrix $M = [\Phi^T\Phi]^{-1}$ is constructed from the data ϕ, and $\hat{\theta}$ is the estimate of θ. Although computationally fast, the amount of data needed for reliable parameter convergence made RLS only slightly faster than a recursive implementation of an *information matrix* form of standard least squares (LS). We have therefore chosen to use such a form of LS which tended to give better estimator performance, traded off against computational burden. In the robotic system application we consider here, payload changes are of a discrete nature at a given point in time, implying that there is no requirement for remembering previous load characteristics. Best results were obtained, then, using a weighted version of the non-recursive expression

$$M^{-1}(k+1)\hat{\theta}(k+1) = \Phi^T(k+1)y(k+1) \tag{3}$$

with recursive data information updates according to

$$M^{-1}(k+1) = \lambda(k)M^{-1}(k) + \phi(k+1)\phi^T(k+1) \tag{4}$$

$$\Phi^T(k+1)y(k+1) = \lambda(k)\Phi^T(k)y(k) + \phi(k+1)y(k+1) \quad . \tag{5}$$

In the above, the weight $\lambda(k)$ is the forgetting factor, and for these applications typically took a value in the range $0.96 - 0.99$.

B. PID Tuning Controller

The concept of automatic tuning for a proportional-integral-derivative (PID) control law has been the subject of recent investigations [14], and in fact has been utilized for many years in flight control systems. Motivation for auto-tuning schemes lies in the fact that often times PID controllers are difficult to tune manually, particularly when a high level of precision is important.

The discrete version of the ideal analog PID controller is given by

$$u(k) = K_P e(k) + \frac{T}{K_I}\sum_{i=0}^{k-1} e(i) + \frac{K_D}{T}[e(k) - e(k-1)] \quad , \tag{6}$$

where T is the sampling time, $u(k)$ is the control input, $e(k)$ is the deviation between the controlled signal and a desired reference, and K_P, K_I, K_D represent the proportional, integral, and derivative

gains, respectively. Generation of a recursive expression for the control input follows easily from (6) as

$$u(k) - u(k-1) = b_o e(k) + b_1 e(k-1) + b_2 e(k-2) \quad . \tag{7}$$

At this point several options are available for choice of the parameters [b_o, b_1, b_2], such as classical pole placement or pole cancellation design. However, for our problem it may not always be apparent *a priori* what the desired closed loop poles should be, since their choice may depend on the effect of payload variation. For example, with a larger payload a slower slew maneuver may be required. For this reason we choose the PID parameters via an optimization of a performance criterion which is *time varying* and which weights the control deviation and the endpoint acceleration.

For the manipulator system of this study it was determined that a performance index which gave adequate results takes the form

$$J(k) = \sum_{k=1}^{q} [k e_s^2(k) + 50000(\Delta u))^2 + 6k^2 \alpha^2(k)] \quad , \tag{8}$$

where $e_s(k)$ represents shaft position error, $\Delta u = u(k) - u(k-1)$, and $\alpha(k)$ is the endpoint acceleration. Notice that this index penalizes endpoint deflections more heavily as time increases, for the following reasoning. In general, when endpoint movement is minimal, the shaft position error term is larger than the acceleration term. Moreover, for a given selection of PID parameters the shaft position error remains virtually the same when a payload is added. However, the endpoint acceleration is noticeably reduced with payload and the relative weight of the square of the acceleration drops. For this system it was observed that endpoint oscillations continue for a relatively long period when a payload is added; this accounts for the k^2 factor in (8). Thus, minimization of (8) reduces the duration of oscillation. A period of 100 samples (T =30 ms) was found to be an adequate interval over which to evaluate the performance index. It is straightforward to derive the relationship between the ARMA model representation (1), the cost criterion (8), and the PID parameterization (6) [5].

The controller design now reduces to computing (8) and carrying out an optimization over possible PID parameters. It is quite obvious, and easily verifiable via experimental tests, that one way to reduce vibrations at the manipulator endpoint when a load is added is to reduce the slew rate. This of course is viable only to a degree since our objective is to attempt to slew as fast as possible with the best possible performance.

Several combinations of proportional, PI, PD, or full PID designs in either or both feedback loops are possible [5]. Here we consider the case for gain adjustment within each feedback path (shaft angle gain and endpoint acceleration gain). The effectiveness of this identification/control scheme is illustrated in Figure 3 for the following profile. First, the arm without payload undergoes a 25° slew with no control applied (open loop); large oscillations are apparent in this first phase. The gains are then tuned and the arm undergoes a further 20° slew in the same direction, then reverses direction for a 45° slew. During this phase the control has been extremely effective in endpoint vibration compensation. In the next phase a 0.415 pound payload is added and the arm undergoes a 20° slew; after five seconds (allowing for damping of deflections) tuning is performed. In the same direction, the arm is then slewed an additional 25°, and in five seconds the direction is reversed to complete the profile with a 45° slew. The two points at which controller tuning was performed are marked on the plot; the total time span for identification and tuning, given the limitations of the laboratory computer, is anywhere from 15 to 25 seconds depending upon the number of parameters tuned in the optimization. For this reason, the time axis in Figure 3 is not marked, but the time period between slew commands is nominally 5 seconds.

IV. Frequency Domain Auto–Tuning Control

A. Identification

An alternative to time domain methods for adaptive filtering is the use of techniques based in the frequency domain [15,16]. Frequency domain adaptive filters enjoy several advantages over their time domain counterparts, including reduced computation and a fast rate of convergence. A disadvantage of methods which identify system frequency response, however, is that auto–tuning (on–line) control design is often at best ad hoc.

As an illustration of one of the major shortcomings of the RLS method, consider the case of identifying the tip acceleration of the one–link apparatus, using zero mean white noise as input. A typical characteristic of time domain methods is the requirement for a persistently exciting input during the identification starting process. This was the case, for example, in [8] where after a significant amount of data was gathered the identifier was turned on and the estimated parameters converged "fast" to the actual ones. That is, although the convergence of RLS is superior to most other time-identification methods, a large amount of iterations is required for convergence to the actual parameters. Shown in Figure 4, the estimated transfer function spectrum of the one–link manipulator is plotted for the cases of 64, 128, 256, and 512 iterations after the start-up of the identification process (30 ms sampling). In all the cases a Butterworth filter of 6th order with a cutoff at $48\frac{rad}{sec}$ was used to prefilter the data, the order of the estimated ARMA system was 5 (these values were found to produce the best results [17]), and all the initial estimated parameters were set to zero. RLS can predict the first mode (at approximately 1 Hz) only after 512 iterations, where the corresponding peak begins to appear. During convergence the estimated poles and zeros of the system were far away from the actual ones. In the case of an abrupt change of the carried payload for the manipulator under consideration, the RLS algorithm needed a significant amount of time to converge to the new system parameters. This characteristic was noticed for the identification and control experiments described in the preceding section, and may not be satisfactory if the control law update scheme is required to respond quickly.

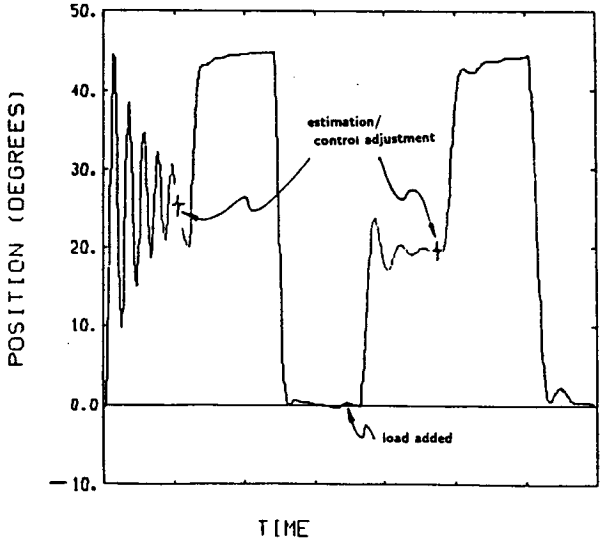


Figure 3: Time Domain Tuning Scheme

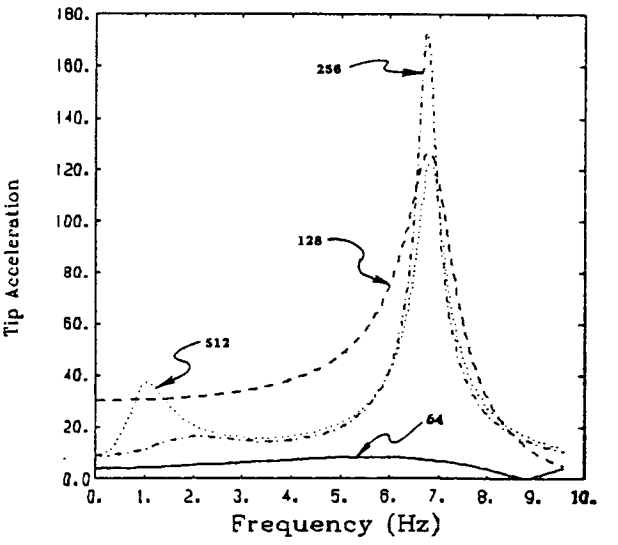


Figure 4: RLS Convergence

By contrast, for on–line filter (transfer function) update, with frequency domain methods the system input signal is transformed to the frequency domain before adaptive filtering is applied. The

simplest frequency-domain adaptive filter is one in which the input signal $u(n)$ and output $y(n)$ are accumulated into buffer memories to form N-point data blocks. These blocks are then transformed by N-point Fast Fourier Transforms (FFTs) to their equivalent frequency transformed blocks U, Y at the k^{th} time instant.

A simple yet effective representation for transfer function identification is the Empirical Transfer Function Estimate (ETFE) [18,19]. A non-recursive updating scheme for the transfer function in the frequency domain, at a given time k, is utilized in the manner

$$H_i(k) = \frac{Y_i(k)}{U_i(k)} \quad , \quad H_{N-i}(k) = H_i^*(k) \quad , \tag{9}$$

for $i \in \{0 \leq i \leq \frac{N}{2}, U(i) \neq 0\}$, where i corresponds to the i^{th} bin in the frequency domain, and $H_i^*(k)$ is the complex conjugate of $H_i(k)$. Notice that the $H_i(k)$'s can be updated every l samples, where $1 \leq l < N$. The main properties of this technique are that: 1) The variance in the estimation is equal to the signal-to-noise ratio at the frequency under consideration; 2) Estimates at different frequencies are uncorrelated (asymptotically as $k \rightarrow \infty$).

A recursive implementation of this idea is possible via the Time–varying Transfer Function Estimation (TTFE) method [20]. The TTFE technique can be used to reduce the variance of the estimated frequency response through two distinguishing characteristics. First, the adjacent frequency bins $H_i(k), H_j(k)$ from Equation (9) are correlated through the relation

$$H_i(k) = \frac{\sum_{j=(i-\Delta_i) \bmod N}^{(i+\Delta_i) \bmod N} \varepsilon_j^i H_j(k)}{\sum_{j=(i-\Delta_i) \bmod N}^{(i+\Delta_i) \bmod N} \varepsilon_j^i} \tag{10}$$

which indicates that the estimate $H_i(k)$ is related to all the adjacent frequencies within a modulus Δ_i with a corresponding weight ε_j^i for the frequency point (bin) ω_j. Notice that the case $\Delta_i = 0$ corresponds to a frequency windowing technique [21] used to reduce the bias and variance of the estimated transfer function. Moreover, the case $\Delta_i = \Delta$ for all i and $\varepsilon_j^i = \varepsilon^i = \Phi_i(u)$ (where $\Phi_i(u)$ is the input spectral density) corresponds to the Blackman-Tukey Procedure [21] for smoothing the estimated transfer function. Therefore, based on this relation the estimated transfer function is a "smoothed" version of the one obtained from ETFE.

The second distinguishing feature of TTFE is that the frequency bin $H_i(k)$ is related to the $H_i(k-1), \ldots, H_i(k-\beta_i)$ bins of the *hybrid* time–frequency domain through

$$H_i(k) = \mathcal{X}[(H_i(k-1), \ldots, H_i(k-\beta_i)] \tag{11}$$

where the function $\mathcal{X}$ may be implemented with RLS for a finite impulse response model $Y_i(k) = \sum_{j=1}^{\beta_i} H_i(k-j)U_i(k-j)$. In case of a sudden change of system dynamics, this recursion results in a smooth transient from the old transfer function to the new one, representing a substantial difference when compared to the nonrecursive ETFE technique which suffers a less smooth transition due to the assumption of orthogonalized input–output data blocks. The computational complexity of TTFE is reasonable and can be decreased by assuming that the frequency bins $H_i(k), H_j(k)$ for the same time instant k are uncorrelated ($\Delta_i = 0$), over a time period of γ samples, where γ corresponds to the updated interval for the adaptation algorithm.

B. Controller Tuning

The critical information for control purposes sought by frequency domain methods is the location of poles and zeroes of the transfer function. These locations correspond to the peaks and the valleys

of the estimated magnitude response. Due to the lightly damped nature of the manipulator these locations are easily recognizable with the TTFE technique, even with signal-to-noise ratios up to $15dB$ [17].

In light of the above discussion on convergence of the parameter estimation, the performance of the TTFE approach in estimating the first modal frequency of the system is demonstrated via experiment. Figure 5 depicts the endpoint acceleration while the arm is slewed through angles of 30° in a square-wave (shaft angle position) reference (8 second period). The arm carries no payload initially, and the ideal frequency is about 1 Hz, indicated by the solid curve. At 8 seconds a 0.485 lb payload (46% of arm weight) is added, and then removed at 16 seconds, resulting in a change in ideal frequency due to the change in payload. Frequency estimates from the TTFE scheme (dashed curve) proved to be adequate, in terms of speed of convergence as well as accuracy, for good controller performance.

Several algorithms for control design using the identified frequency response directly have been implemented for the experimental setup, including a frequency weighted quadratic regulator design [20]. Here we describe results of a design in which the control structure is set within a scheduling framework comprised of two feedback loops: one in which the endpoint acceleration is used as input to a control law, and the other in which the motor shaft angle is input to a separate control law. These two loops are then summed to give a commanded motor input voltage. Motivated by the desire to achieve endpoint position accuracy while maintaining a relatively straightforward implementation structure, simple proportional schemes make up the above-mentioned control laws in the separate loops; it is the individual proportional gains which are scheduled as correlated to frequency domain information over a wide range of payloads. The scheme discussed in the previous section (PID tuning using (8)) was used to establish a "look-up" table for various payloads corresponding to the first modal frequency of the arm. That is, the fundamental frequency (first mode) was found using FFT analysis; because these calculations are carried out off-line, an ample amount of data can be accumulated for the best possible accuracy. The motivation for using the fundamental frequency as the "pointer" in a look-up table of scheduled optimal controller parameters is the obvious relationship with the payload. This fact is exploited in the control law by interpolating four such data points (using four different payloads) in construction of a functional relationship, filling out the look-up table, for use in real-time control.

Figure 6 shows results of the scheduling controller using frequency domain estimation; shown is the arm endpoint position as read by the camera. As before, the gross motion control objective is to track a staircase-shaped reference trajectory. At the end of the first and fourth segment, as indicated, the FFT of the endpoint acceleration is computed and the controller is tuned according to the estimated frequency of the first mode. The first segment is performed in the open loop for a slew angle of 8°; the absence of any control effort is evident from the large overshoot. The controller is then adjusted with the estimated frequency—this operation, including FFT calculation, requires less than 0.3 seconds of CPU on the MicroVax computer. In the next two segments the arm is slewed another 32°, then 40° in the opposite direction; the stabilizing effect of the acceleration feedback controller is evident. A payload of 0.74 lb (69% of total arm weight) is added at the beginning of the fourth segment, as indicated, and the arm is slewed through a commanded angle of 5°. This small level of excitation is sufficient to accurately estimate the first modal frequency, so that the controller is re-tuned to account for the addition of the payload. In the final two segments, the arm is slewed first another 35°, and then 40° in the opposite direction. Again, the control has adequately compensated for deflections at the endpoint. Since the 5° slew with payload generates relatively small deflections, to illustrate the effectiveness of the scheduling control the endpoint position for the case of *not* re-tuning the controller after addition of payload is overlayed in the Figure (dashed line). Similar results were obtained for a variety of payload conditions.

V. Conclusion

This paper has presented a summary of the various identification and control techniques being investigated in the laboratory at Ohio State for flexible-link manipulator systems. Primary attention in these techniques focuses on the ability of the controller to adjust to changes in dynamics, payload, and working environment. Time domain methods offer identified model structures which are readily available for control design, whereas frequency domain methods, particularly the TTFE approach developed for this application, are more desirable when rapid controller tuning is required.

Only a sample of the results obtained to date were presented here due to space constraints; the interested reader is encouraged to pursue the listed references, copies of which are available upon request from the authors.

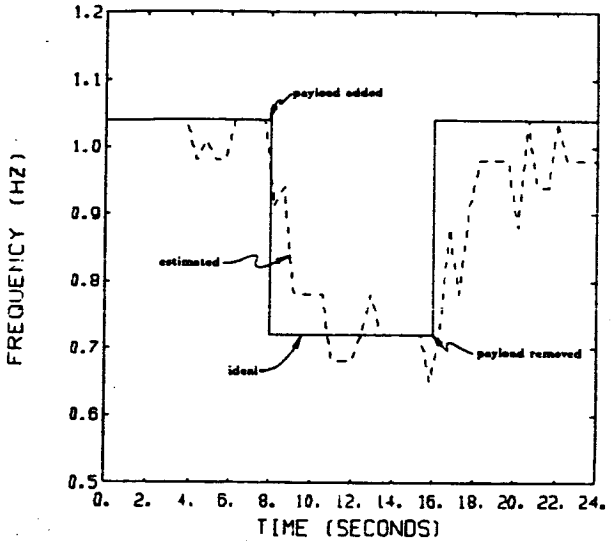


Figure 5: TTFE Results

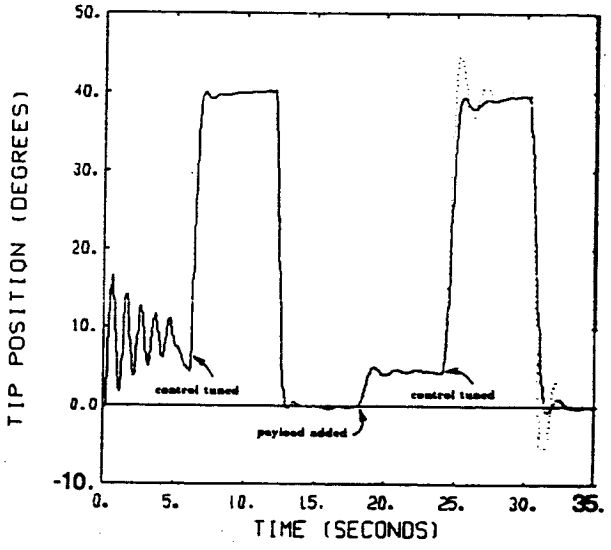


Figure 6: Scheduling Control

References

[1] X. Ding, T. J. Tarn, and a. K. Bejczy, "A novel approach to the modelling and control of flexible robot arms," in *Proceedings of the IEEE Conference on Decision and Control*, Austin, TX, December 1988.

[2] E. Barbieri and Ümit Özgüner, "Unconstrained and constrained mode expansions for a flexible slewing link," *Trans. ASME, J. Dyn., Meas., and Control*, vol. 111, , December 1988.

[3] S. Yurkovich, Ü. Özgüner, A. Tzes, and P. Kotnik, "Flexible manipulator control experiments and analysis," in *Proceedings of the NASA Telerobotics Workshop*, pp. 279–288, January 1987.

[4] P. Kotnik, S. Yurkovich, and U. Özgüner, "Acceleration feedback for control of a flexible manipulator arm," *Journal of Robotic Systems*, vol. 5, no. 3, pp. 181–196, June 1988.

[5] S. Yurkovich and F. E. Pacheco, "On controller tuning for a flexible-link manipulator with varying payload," *Journal of Robotic Systems*, vol. 6, no. 3, , June 1989. (to appear).

[6] S. Yurkovich, F. E. Pacheco, and A. P. Tzes, "On-line frequency domain information for control of a flexible-link robot with varying payload," *IEEE Transactions on Automatic Control*, vol. AC–33, , 1989. (to appear).

[7] R. H. Canon and E. Schmitz, "Initial experiments on the end-point control of a flexible one-link robot," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 62–75, 1984.

[8] D. M. Rovner and R. H. Cannon, "Experiments toward on-line identification and control of a very flexible one-link manipulator," *International Journal of Robotics Research*, vol. 6, no. 4, pp. 3–19, Winter 1987.

[9] D. M. Rovner and G. F. Franklin, "Experiments in load-adaptive control of a very flexible one-link manipulator," *Automatica*, vol. 24, no. 4, pp. 541–548, July 1988.

[10] G. G. Hastings and W. J. Book, "Experiments in optimal control of a flexible arm," in *Proceedings of the 1985 American Control Conference*, pp. 728–729, Boston, MA, June 1985.

[11] G. G. Hastings and W. J. Book, "Verification of a linear dynamic model for flexible robotic manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1024–1029, San Francisco, CA, April 1986.

[12] Ü. Özgüner, S. Yurkovich, J. Martin, and P. Kotnik, "A laboratory facility for flexible structure control experiments," *IEEE Control Systems Magazine*, vol. 8, no. 4, , August 1988.

[13] S. Yurkovich and Ümit Özgüner, "Recent developments in the OSU Flexible Structure Control Laboratory," in *Proceedings of the Seventh VPI & AIAA Symposium on Dynamics and Control of Large Structures*, Blacksburg, VA, May 1989. (to appear).

[14] F. Radke and R. Isermann, "A parameter-adaptive PID-controller with stepwise parameter optimization," *Automatica*, vol. 23, no. 4, pp. 449–457, 1987.

[15] P. J. Parker and R. R. Bitmead, "Adaptive frequency response identification," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 348–353, Los Angeles, December 1987.

[16] M. Dentino, B. Widrow, and J. McCool, "Adaptive filtering in the frequency domain," *Proceedings of the IEEE*, vol. 66, pp. 1658–1659, December 1978.

[17] A. Tzes and S. Yurkovich, "Application and comparison of on-line identification methods for flexible manipulator control," in *Proc. International Conference on Advanced Robotics*, Columbus, OH, 1989. (to appear).

[18] L. Ljung, *System Identification Theory For The User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[19] R. LaMaire, L. Valavani, M. Athans, and G. Stein, "A frequency-domain estimator for use in adaptive control systems," in *Proceedings of the American Control Conference*, pp. 238–244, Minneapolis, MN, June 1987.

[20] A. Tzes and S. Yurkovich, "A new approach to frequency domain identification for flexible structure control," in *Proceedings of the IEEE Conference on Decision and Control*, pp. 1627–1632, Austin, TX, December 1988.

[21] D. R. Brillinger, *Time Series: Data Analysis and Theory*. San Francisco: Holden Day, 1981.

ROBOTIC END-EFFECTORS AND HAND CONTROLLERS

AUTONOMOUS DEXTEROUS END-EFFECTORS FOR SPACE ROBOTICS[1]

George A. Bekey, Thea Iberall, Huan Liu
Computer Science Department
University of Southern California
Los Angeles, California 90089-0782

Abstract

This paper summarizes the development of a knowledge-based controller for the Belgrade/USC robot hand, a five-fingered end effector designed for maximum autonomy. The biological principles of the hand and its architecture are presented. The conceptual and software aspects of the grasp selection system are discussed, including both the effects of the geometry of the target object and the task to be performed. The concluding section of the paper presents some current research issues.

1 Introduction

Grasping and manipulation of objects in space by robotic systems will probably require a blend of teleoperation and autonomy for a number of years. However, the difficulties associated with placement of cameras and other sensors suggest that the robotic end-effectors used in unstructured environments be as autonomous as possible. Our group at USC, in collaboration with the University of Belgrade, has been active for several years in the development of robot hands capable of mimicking some aspects of human prehensile behavior. We have concentrated on autonomous grasping. Hence, the hands we have designed have limited degrees of freedom as required only for grasping and not for finger manipulation. Within this limitation, it is our goal to imbue the control systems for these hands with sufficient intelligence to be able to grasp objects of arbitrary shape with the hand posture appropriate for a given task. This paper presents a brief summary of the major features of the hand design, with emphasis on the software aspects.

[1]This research was supported in part by the Jet Propulsion Laboratory under grant #956501, the National Science Foundation under grants DMC-8719579 and IRI-8796249, and by the Institute for Manufacturing and Automation Research.

2 Human grasping

Following the work of Jeannerod [6], it is known that the human hand preshapes to the geometry of the object being grasped during the approach trajectory. The actual hand posture (grasp mode) selection is accompanied by the selection of the grasp location on the object in such a way as to bring functionally effective forces to bear, insuring a stable grasp appropriate to the task at hand. A model of this process has been developed by Iberall and Arbib [1, 4]. Groups of fingers move generally together as a *virtual finger* setting up the forces that will be applied in opposition to each other. They are functionally effective in the sense that the chosen grasp mode must satisfy multiple constraints acting in the task. A number of investigators have catalogued the basic grasp modes of the human hand [3, 7]. The human perceptual, cognitive and motor systems process geometric information on the target object in the light of the goals of the grasp and with a vast data bank of past experience to obtain the proper grasp mode. In [5], Iberall and MacKenzie identify numerous constraints acting on this process, separating some of the more functional issues from the physical ones. The final configuration of fingers and the applied force are obtained from a blend of sensory feedback and knowledge. We have attempted to incorporate some aspects of this process in the design of our hand.

3 The Belgrade/USC hand

The Belgrade/USC hand is an anthropomorphic, five-fingered end effector. The first model of the hand is illustrated in Fig. 1. It has four articulated fingers and a thumb. The two distal finger joints are not individually controllable; they are connected by linkages in order to move similarly to human fingers during grasping as the fingers flex (a virtual finger). The thumb in Model I was rigid, but capable of rotation about an axis normal to the palm, to bring it into opposition with any of the other fingers. A unique feature of the hand is its autonomous shape adaptation. Three motors are mounted in the wrist structure to provide the external degrees of freedom. One motor moves the thumb, while the others move two fingers each as a virtual finger. The virtual finger drive is applied to each pair of fingers through a lever structure, such that if the motion of one real finger is inhibited, the second can continue to move, thus achieving shape adaptation without external control [2, 13].

The structural design of Model II (to be completed in the Summer of 1989) is illustrated in Fig. 2. This model features a jointed thumb (and hence an additional drive motor) and fingers capable of spreading prior to grasping.

The consequence of this design is that the hand is well suited to autonomous grasping of objects of arbitrary shape; it is capable of preshaping; and is simple to control, since all the motors are located in the wrist structure. Touch sensors are located on the finger tips and on the palm, position sensors are embedded within the fingers. The hand is mounted on a Puma 560 robot.

Figure 1. Belgrade/USC Model I hand.

4 Target geometry and grasp modes

The high-level grasp controller is knowledge-based, selecting a preshape on the basis of visual information on the target and a stored library of relationships between grasp modes and geometric primitives. The basic modules of the system are shown in Fig. 3. A camera provides the input to an image analysis system, which obtains a shape description using generalized cones [12], from which the name and parameters of a geometric primitive are deduced. The system includes such primitives as cone, cylinder, torus, etc. Given the geometric primitive and its dimensions, the system then obtains a list of all feasible grasp modes. This list is unranked; task information is needed to organize it. We have obtained the grasp modes by means of rules and tables [11]. We have also demonstrated a neural network approach to the problem [8, 10].

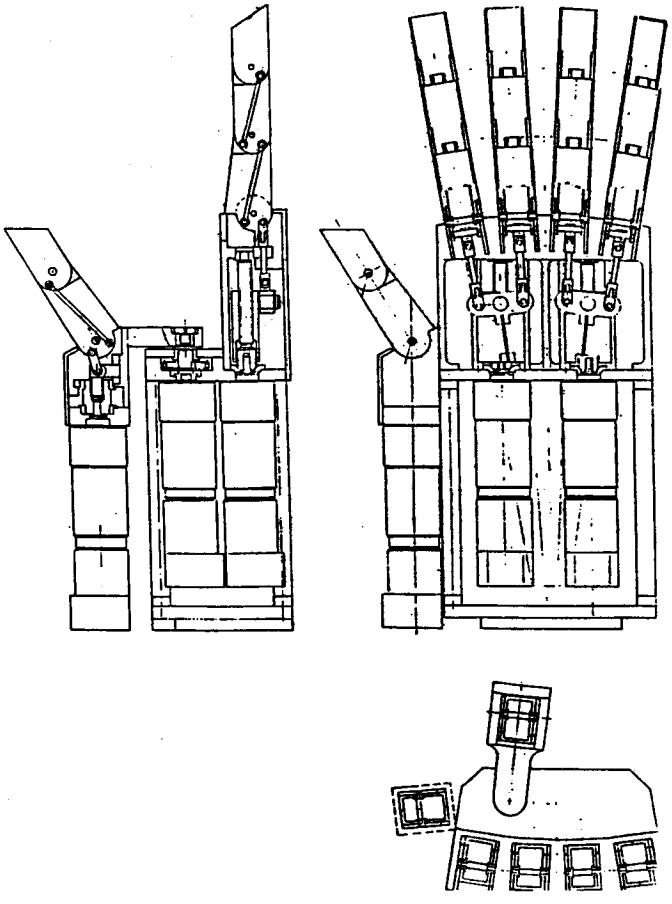


Figure 2. Belgrade-USC Model II hand.

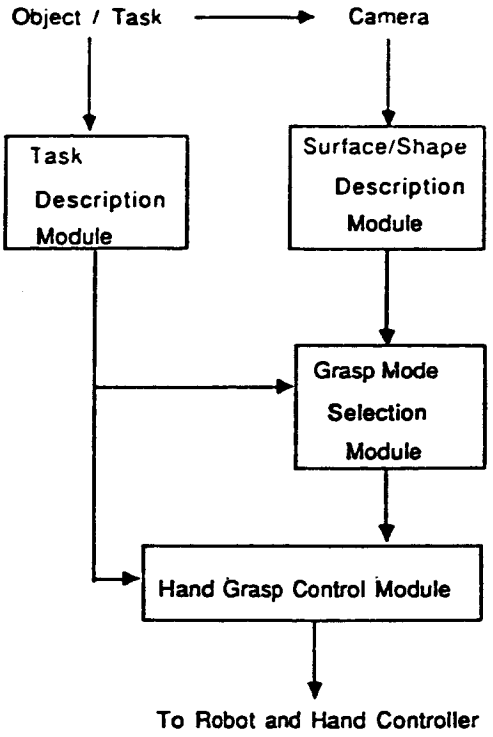


Figure 3. Grasp controller for Belgrade-USC hand.

5 Task information

Our recent work has focused on the question of selecting the preferred grasp mode from among all the permissible ones by using task information [9]. In order to restrict the size of the search space, we have restricted the domain to operations associated with simple assembly tasks, such as:

1. Grasp a wrench to tighten a nut
2. Grasp a hammer to drive the nail.
3. Pull handle to open the drawer.
4. Insert the pin into the hole.

The commands are parsed and the key elements (tool, action, part, context) are extracted and used as inputs to the task analyzer illustrated in Fig. 4. The nature of the action ("turn", "insert", etc.) and the type of tool are used with a functional database to determine the focus of the desired action. For example, using a wrench puts the focus on the ability to apply the maximum possible torque; inserting a pin requires the greatest possible ability to manipulate a small object in space; pick and place operations require a highly stable grasp. These action

foci in turn make it possible to select which of a large number of heuristics concerning human grasping are appropriate to the task. Humans use such heuristics as "grasp the object as close to the center of gravity as possible" for some tasks and "grasp the object near the end" for others. Using a wrench to tighten a nut requires a different grasp heuristic than picking up the wrench to place it in a given location. As illustrated in Fig. 4, once the heuristics are selected and ordered, they are used to produce a rank-ordered list of grasp modes. The highest ranked mode is selected and the hand is preshaped accordingly. The details of this process are described in [9].

Currently our object analyzer contains descriptions of 5 primitives (cylinder, cube, torus, sphere and cone). The functional object database contains descriptions of 7 objects (wrench, screwdriver, hammer, nut, pin, handle and cylinder) as well as type of tool, center of gravity, function and other information. The task analyzer contains 72 production rules, 14 heuristics and 2 meta-heuristics (used to order the heuristics). The Model I hand is capable of 4 grasp modes (power grasp, hook grip, pulp pinch and lateral pinch). Additional modes usually associated with the human hand will be possible with Model II.

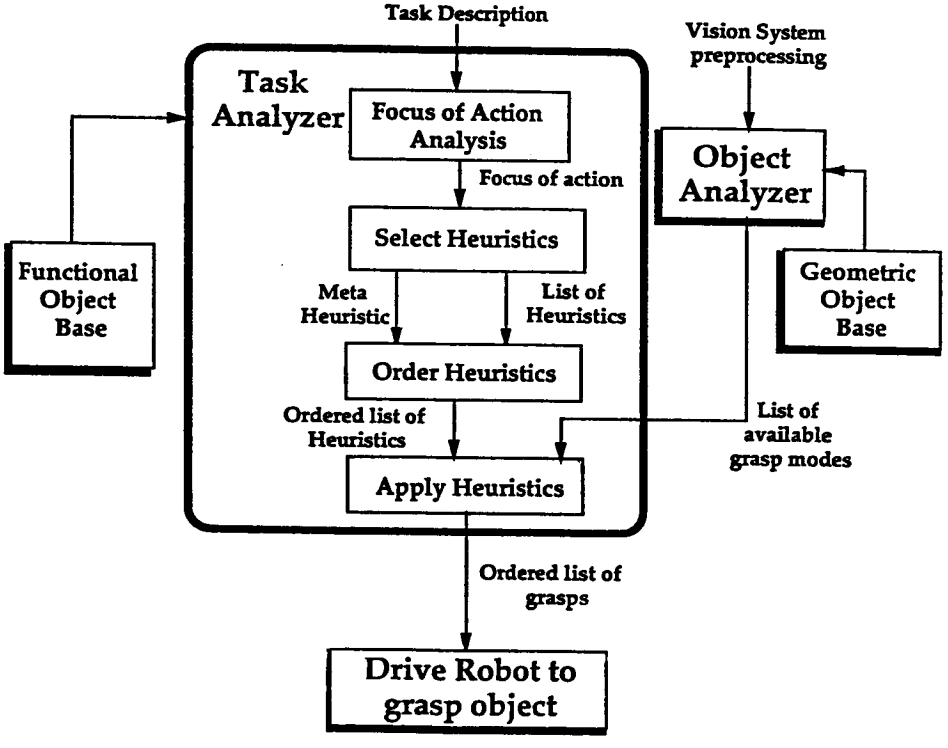


Figure 4. Task analyzer.

6 Current and future work

We have developed a knowledge based approach to reasoning about grasping from a task description, and used it succesfully to obtain grasp modes for a robot hand. The attributes of the task and knowledge about actions, objects and geometry have enabled us to rank order feasible grasps in order of quality.

Much work remains to be done to extend the functional object base and to broaden the task descriptions and attributes. We also plan to add slippage sensing to the hand and to include surface friction and estimated object weight into the system.

We believe that the approach to autonomous grasping described here, where both task and object geometry are considered in the determination of a grasp, offers great potential for space applications.

References

[1] Arbib, M. A., Iberall, T. and Lyons, D. (1985): Coordinated Control Programs for Movements of the Hand. In: *Hand Function and the Neocortex*, A. W. Goodwin and I. Darian-Smith (Eds), Berlin: Springer Verlag, 111-129.

[2] Bekey, G.A., Tomovic, R., and Zeljkovic, I. Control Architecture for the Belgrade/USC Hand. In: S.T. Venkataraman and T. Iberall (eds) *Dextrous Robot Hands*, Springer-Verlag; in press.

[3] Cutkosky, M. R. and Wright, P.K. (1986): Modeling Manufacturing Grips and Correlations with the Design of Robotic Hands. *Proc 1986 IEEE International Conf on Robotics and Automation*, San Francisco, Calif, April, 1533-1539.

[4] Iberall, T., Bingham, G. and Arbib, M. A. (1986): Opposition Space as a Structuring Concept for the Analysis of Skilled Hand Movements. In: *Generation and Modulation of Action Patterns*, H. Heuer and C. Fromm, (Eds), Berlin: Springer-Verlag, 158-173.

[5] Iberall, T. and MacKenzie, C. L. Opposition Space and Human Prehension. In: S.T. Venkataraman and T. Iberall (eds) *Dextrous Robot Hands*, Springer-Verlag, in press.

[6] Jeannerod, M. (1981): Intersegmental coordination During Reaching at Natural Visual Objects. In: *Attention and Performance IX*, J. Long and A. Baddeley (Eds), Hillsdale: Erlbaum, 153-168.

[7] Lister, G. (1977): *The Hand: Diagnosis and Indications.* Churchill Livingston, London.

[8] Liu, H., Iberall, T., and Bekey, G.A.(1988): Building a Generic Architecture for Robot Hand Control, *Proc 1988 IEEE Conference on Neural Networks*, San Diego, Calif, July 24-27, 567-574.

[9] Liu, H., Iberall, T., and Bekey, G.A.(1988): Reasoning about Grasping from Task Descriptions, *Conf. Intell. Robotics and Computer Vision, SPIE Proceedings, vol 1022*, Cambridge, Mass, November 7-11.

[10] Liu, H., Iberall, T., and Bekey, G.A. Neural Network Architecture for Robot Hand Control, *IEEE Control Systems Magazine*, in press.

[11] Liu, H., Iberall, T., and Bekey, G.A.(1989): The Multi-dimensional Quality of Task Requirements for Dextrous Robot Hand Control, *Proc 1989 IEEE Conference on Robotics and Automation*, Scottsdale, Arizonia, May 14-19.

[12] Rao, K., Medioni, G., Liu, H. and Bekey, G.A. (1989): Shape Description and Grasping for Robot Hand-Eye Coordination, *IEEE Control Systems Magazine*, 9(2): 22-29.

[13] Tomovic, R., Bekey, G.A., and Karplus, W.J.(1987): A Strategy for Grasp Synthesis with Multifingered Robot Hands, *Proc 1987 IEEE Conference on Robotics and Automation*, Raleigh, NC, March 30-April 3, 83-89.

DESIGN AND CONTROL OF A MULTI-FINGERED ROBOT HAND PROVIDED WITH TACTILE FEEDBACK

H. Van Brussel, B. Santoso, D. Reynaerts

Katholieke Universiteit Leuven
Department of Mechanical Engineering
Celestijnenlaan 300B
B-3030 Leuven, Belgium

Abstract

The design, construction, control and application of a three finger, nine degrees of freedom robot hand, with built-in multi-component force sensors are described. The adopted gripper kinematics are justified and optimized with respect to grasping and manipulation flexibility. The construction features miniature DC-motor drive systems imbedded into the fingers. The control is hierarchically structured and is implemented on a simple PC-AT computer. The hand's dexterity and "intelligence" are demonstrated with some experiments.

1. Introduction

The fascinating dexterity and versatility of the human hand caused many people to dream about the development of a mechanical equivalent of their own hands. For about fifteen years, researchers in the whole world [1],[2],[3],[4],[5],[6] are challenging this problem and yet their results seem to be rather poor in comparison with the natural example. On the other side, when looking at the present day two jaw industrial grippers, the developed multifingered grippers really are a big step forward, without being copies of human hands.

Generally spoken a robot end effector or gripper has two functions. In the first place it should be able of grasping a wide variety of objects in order to augment the versatility of the robot on which it is used. On the other hand, it should be able to perform short-range manipulations without the necessity to move the whole robot arm, thereby providing local redundancy. Especially in assembly tasks these fine manipulations can be very useful.

Our aim at K.U.Leuven was not to build an equivalent of the human hand, but rather to construct a dextrous end effector providing both grasping and manipulating functions [7]. Unlike most other designs, the main effort was given to the manipulating function. The mechanical design is not optimized with respect to weight requirements, as our first intention was to demonstrate that a multifingered gripper provided with force sensor feedback, even when using a rather small controlling computer, really can perform the desired manipulative dexterity.

2. Kinematic consideratons

2.1. Basic presumptions

When designing a multifingered gripper the number of fingers to be used is the first problem to cope with. For every finger also a suited layout has to be chosen. By the formulation of a minimization function one can find a solution for this problem. As described by Salisbury and Craig [8], the contact between an object and a finger can be classified by the number of degrees of freedom (d.o.f.) of relative motion it permits. It is evident that the number of d.o.f. is inversely related to the number of constraints on object motion (c.o.m). For every type of contact, the numbers of constraints and degrees of freedom are listed in table 1. The term *soft finger* is used to denote a contact area with enough friction to resist moments about the contact normal.

Table 1 : Contacts between object and finger

Type of contact	Symbol	c.o.m.	d.o.f.
Planar contact with friction	x_6	6	0
Line contact with friction	x_5	5	1
Soft finger	x_4	4	2
Point contact with friction or planar contact without friction	x_3	3	3
Line contact without frition	x_2	2	4
Point contact without friction	x_1	1	5

Further, one could define *active joints* and *passive joints* [7]. An active joint is a joint where the relative positions of the two links can be set by external means. An example of an active joint is a servoed joint. A passive joint is a joint where the relative positions of the two links is depending on constraints imposed by the kinematic linkage.

In our design, we assume a *fingertip-type prehension* of the object. This kind of prehension, where every finger has only one contact with the object, is only one of the six possible types of hand prehension [9]. It was chosen because of its excellent moving capabilities, which was, as stated before, considered more important than the lack of performance when speaking in terms of grasping. Furthermore it is simplifying considerably the control of the gripper.

2.2. Kinematic criteria

Starting with these presumptions one can formulate some kinematic criteria that have to be met by every hand design based on a fingertip prehension:

- To be able to move the object in n degrees of freedom, a minimum of n degrees of freedom is needed at each finger-object linkage. So when the contact has 1 d.o.f. the connecting linkage needs n-1 d.o.f.

- Each finger has to be able to reach the required contact point, so the minimum number of active joints will be equal to the dimensionality of the object.
- To be able to completely restrain a three-dimensional object from motion, the minimum number of restrictions of all contact points is six.
- The contact between object and finger is not a permanent contact. Therefore at least one additional connectivity restriction has to be added.

2.3. Minimization of the number of active joints

To facilitate the control of the gripper, one could minimize the number z of active joints in the system :

$$\text{Min } z = 3x_1 + 3x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6, \quad (1)$$

where x_i (i=1,...6) are defined in table 1. The problem is subjected to following conditions:

$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \geq 7 \quad (2)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 2 \quad (3)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 5 \quad (4)$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \text{ and integer} \quad (5)$$

Condition (2) means that we need at least seven (six for mobility and one for connectivity) constraints. When only using point contacts without friction, this condition becomes:

$$x_1 \geq 7$$

This is the same condition as stated by Lakshminarayama [10]. Conditions (3) and (4) are expressing that a hand should have at least two and at most five fingers. They are based on the fact that every finger has only one contact with the object.

The problem (1) as described above is a linear integer programming problem that can be solved by the simplex method using Gomory's algorithm. This yields as optimal solution :

$$z = 7 \qquad x_1 = 0 \quad x_2 = 0 \quad x_3 = 1$$
$$x_4 = 1 \quad x_5 = 0 \quad x_6 = 0$$

So a two-fingered hand with at one finger a point contact with friction and at the second finger a soft finger contact is using the smallest possible number of active joints namely seven. The drawback of this gripper is its limited ability to resist moment about the line connecting the contact points and also its limited mobility.
To improve mobility one could replace the point contact by a plane contact with friction which requires two extra active joints. Because the surface geometry of the object has to match the surface geometry of the plane, this solution lacks universality.

The next choice is to build a robot hand with three fingers. Thus changing the right hand side of equation (3) yields following results:

$$z = 9 \qquad x_1 = 1 \quad x_3 = 2 \quad x_2, x_4, x_5, x_6 = 0$$

Because a point contact without friction is a rather academic concept, the

finally constructed hand has three fingers, each finger having three active joints and a point contact with friction at the fingertip.

3. Mechanical design

3.1. Configuration

When designing a finger with three active joints, several combinations of rotational and translational joints are possible. Translational joints would imply a very complicated construction, so we preferred to use only rotational joints. Even when using only rotational joints, still a wide variety of possible finger designs exists, and there seems to be no evident criterion to make a selection. Therefore only the three configurations, described in figure 1 and mainly inspired on the human finger, are further considered.

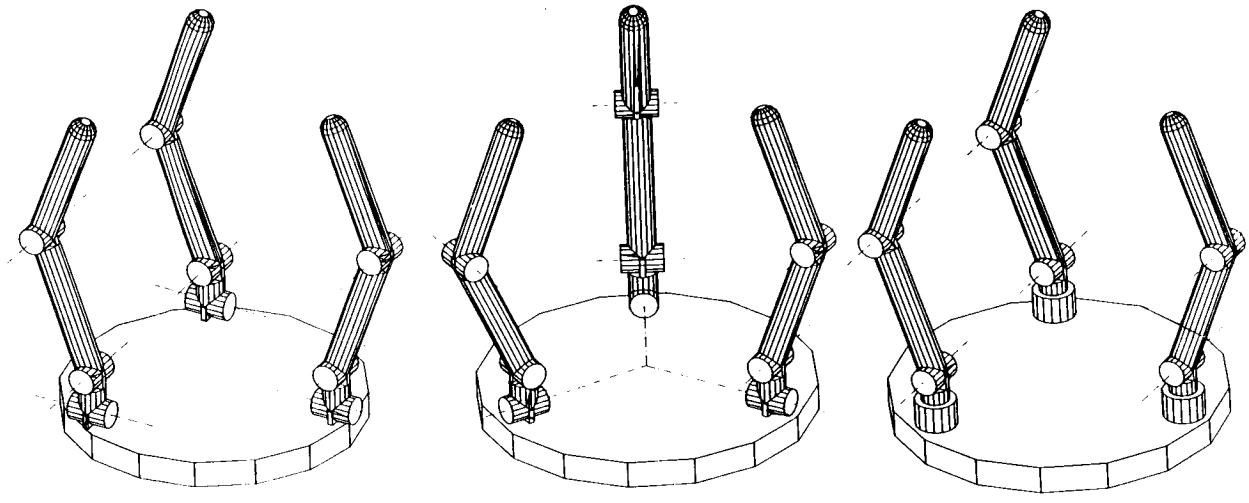


Fig. 1. Possible hand configurations

To make the final choice, we studied these designs for their grasping ablility of some simple geometric forms :

- A square object is grasped with the fingers working as a two-jaw parallel gripper. When the finger shape is cylindrical all three configurations can perform this task.
- For a vertical cylindrical object, the first configuration cannot make a line contact for all fingers. The other two configurations are equivalent.
- When grasping a small horizontal cylinder the fingers of the second design may touch each other so that only the third design remains.

This configuration has a drawback for manipulating the object, because of a singularity in the working space at the rotating axis of the first joint. Therefore the working space had to be limited.

3.2. Construction

To simplify the construction of the first prototype, we designed fingers with built-in electrical actuation, rather than using a remote tendon type actuation. Electrical actuation was also very attractive because of the ease of control. So the fingers are constructed from rotational joints driven by dc motors, using planetary reduction gears to generate an acceptable torque. Some motor parameters (gearbox included) are:

dimensions	24 x 60 mm
weight	1.2 N
max. speed	6.8 rps
max. torque	2.3 Nm
gear ratio	1 : 1164

A miniature incremental position encoder (125 pulses/rev.) is attached to the axis of every motor. Due to the high transmission ratio, this results in a very high resolution of the position measurement (145500 pulses/ finger rev.). Every finger is also equipped with a three dimensional force sensor using strain gauges. These force sensors were built as a combination of one ring dynamometer and two cantilever boxes [7]. The characteristics of the sensors are :

maximum force	50 N
resolution	0.2 N
nonlinearity	< 1 % full scale
average cross sensitivity	< 10 %
acquisition speed	20 kHz
bandwidth	25 Hz (determined by the filters)
drift	10 % full scale in 4 h

The final design of the three-fingered gripper can be seen in figure 2. On the photograph the gripper is mounted on a fixed structure, grasping a chicken egg. The mechanical size of the fingers is determined by the size of the components. Every fingertip is equipped with a rubber ball to introduce friction at the contact points. The force sensors built-in in the first phalanx are clearly distinguishable. The amplifiers for the strain gauge signals are placed on the bottomplate of the gripper together with all other connectors for the motors and the encoders.

4. Controller design

4.1. Mathematical gripping model

For real applications the radii of the contact surfaces may not be too small in order to limit the contact pressure. This means that a point contact becomes a ball contact which makes the kinematic relations between finger and object much more complicated. The use of a soft layer at the contact point will create a second problem. When using a soft layer, a force tangential to the contact area will shift the center of a ball contact. As a consequence the contact point will shift when there is a rotation around the normal line. One can conclude that the calculation of the exact kinematic relations becomes very difficult in practical applications so that there has to be some possibility to compensate for calculation errors.

One of the solutions is to have compliances at the finger tips in order to compensate for position errors. This method is also very useful to compensate for small fingertip deviations caused by the controller itself. Otherwise the coordinated movement of the three fingers would require a very complicated servo system. The proposed gripping method simulates a three dimensional spring behaviour at finger tip. As a consequence the position accuracy of the object relative to the gripper base becomes rather low. However, from extensive experience with active force feedback at K.U.Leuven [11], a robot with a compliance at the end effector is still able to do accurate operations when the external forces working on the object are measured and fed back to the controller.

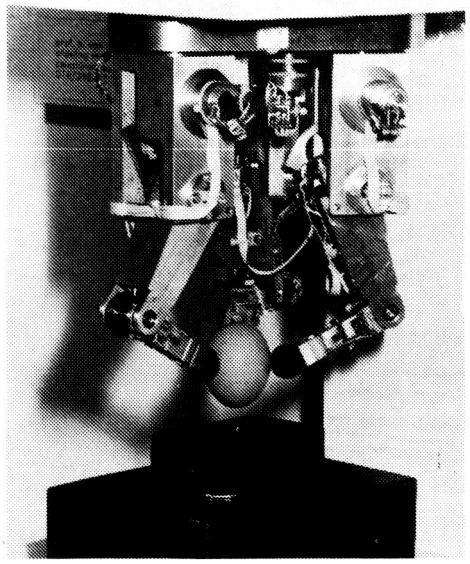


Fig. 2. General layout of the dextrous gripper

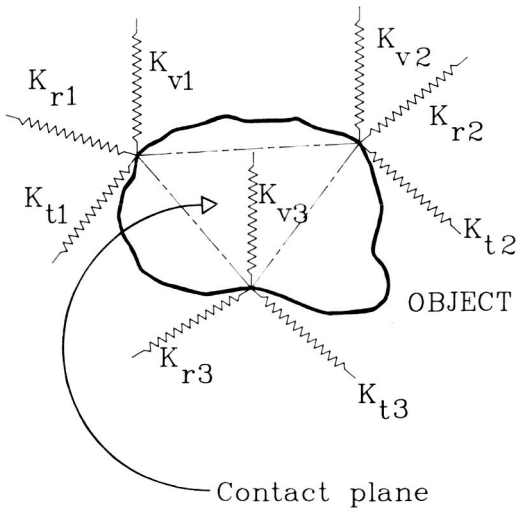


Fig. 3. Contact plane definition

When the object is held by three sets of three linear springs working through the contact points, the contact plane is defined as the plane passing through these three points (Figure 3). The object can be moved relative to the gripper base by moving the contact plane. The positions of the contact points and springs relative to the contact plane remain the same.

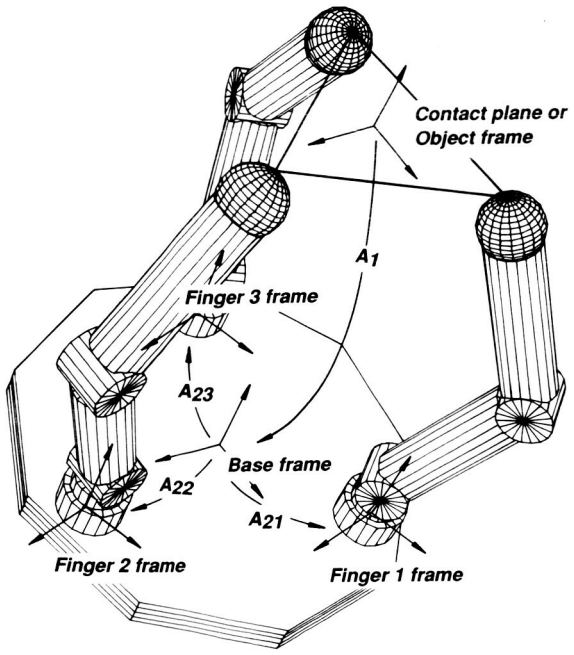


Fig. 4. Coordinate frame definitions for transformations from object frame to finger joints

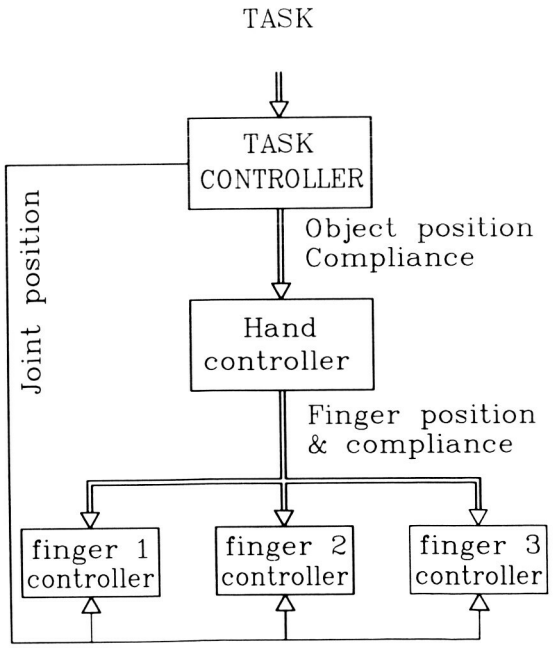


Fig. 5. Control hierarchy for dextrous gripper

To calculate the joint positions of each finger, the coordinates of each finger tip in the contact plane relative to the object frame, are transformed to the hand base frame by means of transformation matrix A_1 (Figure 4) and to the finger base frames through transformations A_{2i} (i=1,2,3). To move the object, transformation matrix A_1 is calculated for the successive positions.

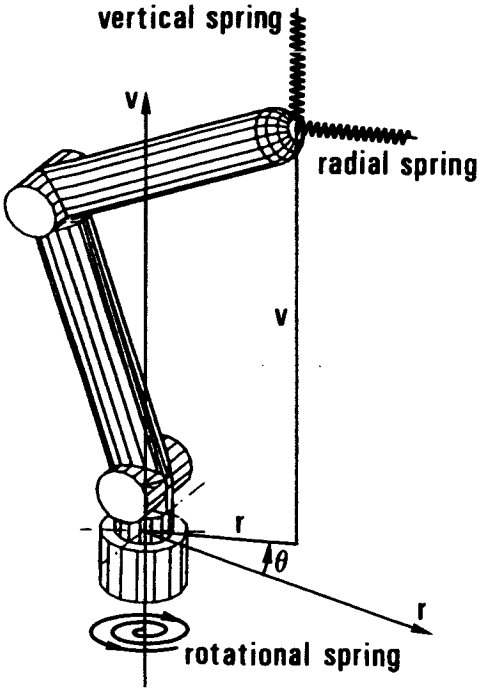
4.2. General controller structure

The controller is hierarchically divided into three main levels: the finger controller, the hand controller and the task controller.
Normally the hand level has the control over the finger level. As can be seen on figure 5, there is some special case where the joint positions are almost directly routed from task level to finger level. This happens when the fingers have no contact with the object and are following a preprogrammed approach path in order to grasp the object. The precise working of this case is discussed further.

4.3. The finger controller

Basically each finger of the robot hand is controlled as an active stiffness system, where the finger tips are programmed as two linear springs (vertical and radial) and one rotational spring. The fingers are thus controlled in cylindrical coordinates as shown in figure 6. In fact this is a simplification of the finger model described above (three cartesian springs attached to the object). These simplifications were made to reduce the required computing power. The inputs to the finger controller, coming from the hand controller, are :

- spring rates for radial,vertical and rotational springs;
- vertical, radial and rotational position of the finger tip relative to the finger frame.



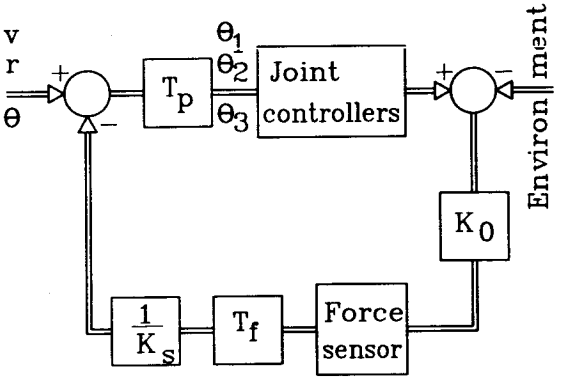


Fig. 7. Finger controller layout

Fig. 6. Cylindrical coordinates used for spring definition

The software controlled spring is realized by using a multidimensional stiffness controller with internal position loop. The internal position loop scheme was chosen mainly because of the higher bandwidth of the position measurement in comparison with the force measurement. The additional passive compliance which is usually needed for an internal position loop was unnecessary because of the high resolution of the position measurement [11].

Figure 7 shows that the cylindrical finger coordinates (v,r,θ) are tranformed to joint coordinates $(\theta_1\theta_2\theta_3)$ for the joint position controllers. The new position results in an external force (determined by the contact stiffness K_O), measured by the strain gauges. This signal is passed trough an analog low pass filter before it is converted into a digital signal. The force measurements are transformed by T_f into the cylindrical coordinate system of each finger. They give the measured forces in radial and vertical direction and a moment around the rotational axis. Both moment and force are multiplied by the desired spring compliance $1/K_S$. The result is subtracted from the desired position.
Digital PD-controllers are used to control the finger joints. Velocity feedback is calculated from the position by differentiation of the encoder signal, as there was no place to use a tachometer.

4.4. The hand controller

The task of the hand controller is the coordination of the position of all fingers. One has to make a distinction between the grasping and the manipulation of an object.
During the grasping (before there is contact between the fingers and the object), the object does not move and the position of the contact plane relative to the robot hand base remains unchanged, while, however, the positions of the finger tips relative to the contact plane or object frame are changing. This seems somewhat contradictory because of the fact that the contact plane was defined as a plane formed by the fingertips. But this definition was only valid when there is a contact between finger and object. In this case the con-tact plane is formed by three arbitrarily predefined points on the object where the gripper is going to grasp it. So one can have the impression that the fingers are direcly controlled by the task level as was indicated in figure 5.
A second phase is the manipulation of the object relative to the hand base. As explained above, this is done by moving the contact plane, formed by the contact points. The position of the finger tips relative to the contact plane remains unchanged. The movement of the contact plane consists of a rotation and a translation relative to the hand base. The rotation transformation is using the RPY angles formalism.

4.5. The task controller

The task controller is controlling the performed operations. This controller is written in a high level language unlike the two previous controllers that are written in assembly language. It is providing a set of subroutines which make it very easy for the programmer to develop a specific application program.
We take as an example a task that requires the robot hand to grip an object, move it in vertical direction and then move it back until it is touching the base. Figure 8 gives the Fortran program performing this task.

```
c gripping the object
      do 100 contactforce=0
      touch1
      touch2
      touch3
100   continue
c move in vertical direction
      moveinz (disp)
c move down until vertical force equals
c "touchforce"
200   moveinz (-1)
      getforce (fz)
      if ( fz < touchforce) then
      go to 200
      return
      end
```

Fig. 8. Sample program of grasping task definitions

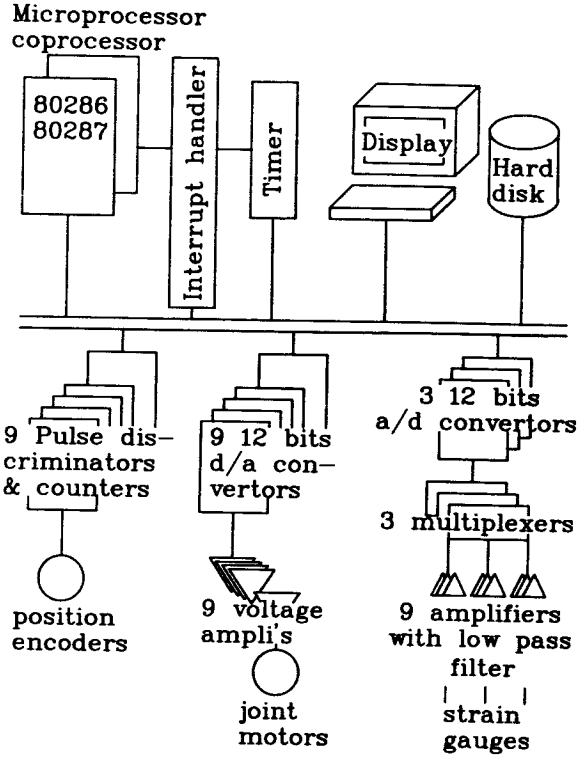


Fig. 9. Hardware configurations of the hand controller

4.6. Hardware construction

Figure 9 gives a description of the hardware implementation of this controller as it was developed in 1986. The controller is based on the Intel 80286 microprocessor with 80287 numerical coprocessor, both running at 9 Mhz. This system has 512 kbyte ram memory,20 mbtye mass-storage, a timer and an interrupt handler. The interfaces built to connect the robot hand to the controller are :

- decoder inputs for the incremental optical position encoders.
- digital to analog converters (12 bits) with power amplifiers to drive the servomotors
- analog to digital convertors for measuring the outputs from the force sensors

As the floating point operation speed of the controller was insufficient to perform all coordinate transformations, some calculations had to be made by using integer arithmetic, resulting of course in a decreasied accuracy of all mathematical operations.

5. Experiments

5.1. Manipulation of an object

This experiment was set up to prove the ability of the gripper to absorb the error between calculated and real position of the contact points. It also demonstrates that it is possible to manipulate objects having a complicated and only approximately known geometry. The setup of this experiment is shown in figure 2. The fingertip was assumed being a point and no correction was

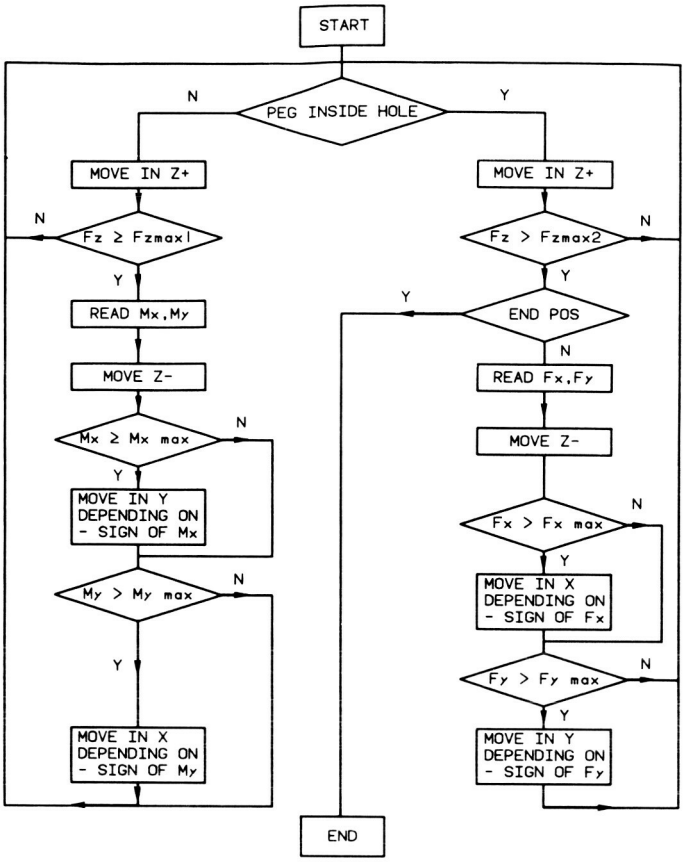


Fig. 10. Flow chart of insertions task

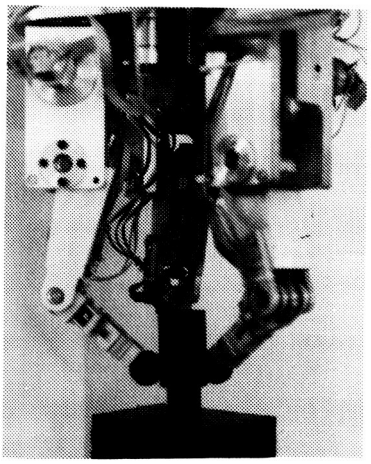


Fig. 11. Peg-into-hole insertions setup

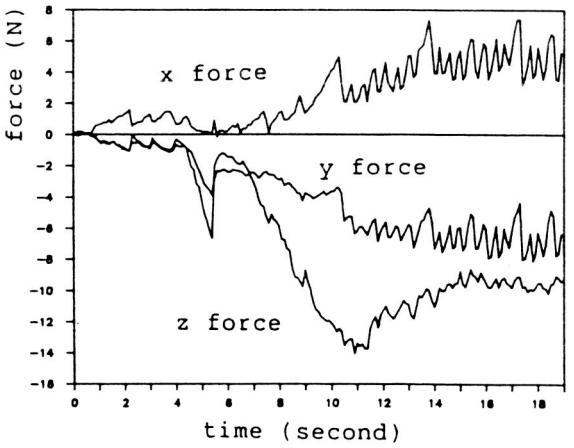


Fig. 12. Force history during insertion

made for the shifting between fingertip and egg. In a first step the gripper will grasp the egg by approaching the fingers until there is a contact force detected. Once there is contact, the egg can be translated or rotated in any direction in space. Of course every movement is subject to the limited workspace of the gripper.

5.2. Peg into hole insertion

By using a chamfered peg and an adapted compliance one can eliminate centering errors during an insertion [12]. This method works well if the initial displacement error is smaller than the dimension of the chamfer.
Our experiment (Figure 11) uses an unchamfered peg to be inserted in an unchamfered hole with a clearance of 0.1 mm. The use of a searching algorithm makes insertion possible even with an initial displacement error of one fourth of the diameter of the hole. The flow chart for the algoritm is shown in figure 10. If the peg is not precisely centered when starting the insertion, the force in the vertical direction increases until it reaches a maximum preset level. The program then calculates the forces in both horizontal directions to find out the magnitude and direction of the moment acting on the peg because of the eccentric insertion. The object will be retracted until the vertical force equals zero and then will be moved to another position following the direction of the calculated moment. This procedure will be repeated until the vertical movement surpasses a preset distance after which the peg is to be considered as rightly centered inside the hole. During the last part of the insertion,some adjustments of the peg position will be carried out in order to keep the horizontal forces as low as possible.

The forces working on the object during the inserton can be seen in figure 13. Both horizontal directions correspond to x and y; z is the vertical insertion direction.

6. Further developments

The first aim of this project was to build a dextrous multifingered gripper with extensive grasping and manipulative capabilities, with a limited computer budget, as is common for space applications. Therefore no attempt was made at this stage to optimize the mechanical design.

The next step is to improve the design to arrive at a more technically sound concept. Therefore the volume and weight need to be reduced and the actuator capabilities increased. The next design will not only allow a three fingered grasp, but also other types, like e.g. a planar grasp [9]. This implies another type of force sensing.

The first problem was to find a suitable actuator. A theoretical study [13] comparing electric, hydraulic, pneumatic and shape memory actuation, proved electric actuation to be the most appropriate solution, especially when taking into account the development time and costs. Parallel developments in other labs showed also the need for a remote tendon type actuation. So we decided to develop a cable pulley actuation driven by a linear electric actuator. As there was not a suitable commercial device available for our application, a new actuator was designed. Figure 13 shows a cross-section of this device. It is built up around an Inland frameless high torque rare earth DC motor and uses a miniature high precision ball screw

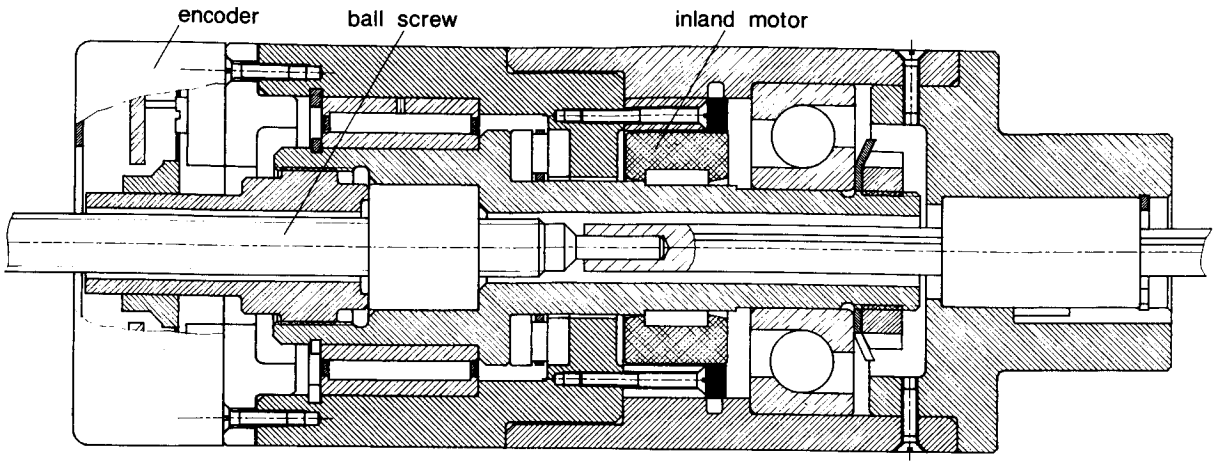


Fig. 13. Linear actuator for controlling tendon-controlled finger

to obain a linear movement. The actuator is designed to reach a peak force of 300 N with a linear stroke of 30 mm. A first prototype is presently being tested.
Also a first model of a finger with cable-pulley actuation is built. We designed a three joint finger driven by four cables, three for flexion and one for extension. It is intended to equip this device with an already developed tactile sensor based on conductive rubber [14].

7. Conclusion

A universal gripper should have two functions : grasping and manipulating an arbitrary object. The main effort in this study was given to the manipulating function. By minimization of the number of active joints, is was found that a three fingered hand with in total nine active joints is the most optimal design in the case of a three fingertip grasp.
Active compliance has proven to be an effective solution for certain problems, occuring during the manipulation of an object by a multifingered gripper. The practical applicability of this method was demonstrated with a simple multifingered gripper, controlled by an ordinary personal computer. Actually this gripper is redesigned to optimize its mechanical construction and to extend of its gripping capabilities. The final aim is not to build an equivalent of the human hand, but rather to construct an industrial end effector providing both dextrous grasping and moving functions.

8. References

[1] JACOBSEN S.C., IVERSEN E.K., KNUTTI D.F., JOHNSON R.T., BIGGERS K.B.,1986, Design of the Utah/MIT dextrous hand, Proc IEEE Int. Conf. on Robotics and Automation.
[2] DARIO P., BUTTAZZO G., 1987, An anthropomorphic robot finger for investigating artificial tactile perception, The International Journal of Robotics Research Vol 6 No 3.
[3] NAKANO Y., FUJIE M., HOSADA Y., 1984, Hitachi's robot hand, Robotics age.
[4] ASADA H., HANAFUSA H., 1977, Stable prehension by a robot hand with elastic fingers, Proceedings 7th ISIR, Bedford (UK): IFS.

[5] ROVETTA A., CASARICO G., 1978, On the prehension of a robot mechanical hand : theoretical analysis and experimental tests, Proceedings 8th ISIR, Bedford (UK): IFS.
[6] MASON M.T., SALISBURY J.K., 1985, Robot hands and the mechanics of manipulation, MIT Press.
[7] SANTOSO B., 1987, Design and control of a multifingered robot hand with tactile feedback, PhD Thesis 87D4 Dept. Mech. Eng., K.U.Leuven (Belgiu).
[8] SALISBURY J.K., CRAIG J.J., 1982, Articulated hands : Force control and kinematic issues, The International Journal of Robotics Research, Vol 1. No 1.
[9] TANIE K., 1985, Design of robot hands, Handbook of industrial robots, John Wiley.
[10] LAKSHIMINARAYAMA K., ASME Paper, No 78 det 32, 1978.
[11] DE SCHUTTER J., 1986, Compliant motion : Task Formulation and Control, PhD Thesis 86D1 Dept. Mech. Eng., K.U.Leuven.
[12] DRAKE S.H., WATSON P.C., SIMUNOVIC S.N., 1977, High speed robot assembly of precision parts using compliance in lieu of sensory feedback, Proc 7th ISIR Tokyo.
[13] [illegible]AERTS D., 1987, Technical report 2142 : Selection of drives for robot applications, Esprit Project 1561 : Sacody, K.U.Leuven (Belgium).
[14] VAN BRUSSEL H., BELIEN H., BAO C.-Y., 1989, Force/torque and tactile sensors for sensor-based manipulator control, In Proc. 1989 Nasa Conf. on Space Telerobotics, Jet Propulsion Laboratory.

TRACTION-DRIVE FORCE TRANSMISSION FOR TELEROBOTIC JOINTS*

D. P. Kuban and D. M. Williams

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6353

Abstract

The U.S. Space Station Program is providing many technological developments to meet the increasing demands of designing such a facility. One of the key areas of research is that of telerobotics for space station assembly and maintenance. Initial implementation will be teleoperated, but long-term plans call for autonomous robotics. One of the essential components for making this transition successful is the manipulator joint mechanism.

Historically, teleoperated manipulators and industrial robotics have had very different mechanisms for force transmission. This is because the design objectives are almost mutually exclusive. A teleoperator must have very low friction and inertia to minimize operator fatigue; backlash and stiffness are of secondary concern. A robot, however, must have minimum backlash, and high stiffness for accurate and rapid positioning. A joint mechanism has yet to be developed that can optimize these divergent performance objectives.

A joint mechanism that approaches this optimal performance was developed for NASA Langley Research Center, Automation Technology Branch. It is a traction-drive differential that uses variable preload mechanisms. The differential provides compact, dexterous motion range with a torque density similar to geared systems. The traction drive offers high stiffness and zero backlash – for good robotic performance, and the variable-loading mechanism (VLM) minimizes the drive-train friction – for improved teleoperation. As a result, this combination provides a mechanism to allow advanced manipulation with either teleoperated control or autonomous robotic operation. This paper will address the design principles of both of these major components of the joint mechanism. Various materials were evaluated for the traction rollers, and two were tested. Also, various surface modifications to these rollers were studied utilizing previous NASA Lewis Research Center experience. Both modified and unmodified materials were tested. For the VLM, several designs were investigated to determine the trade-offs between friction and compliance, as well as the effects of dimensional tolerances and structural deflection. Various designs were fabricated and tested. Test results from the test joints are included. Also, the preliminary results of the complete master/slave assembly are discussed. At the time of this writing, final assembly is under way. Finally, the paper describes some of the limitations of this mechanism, as well as recommendations for further development of this technology.

*Research sponsored by NASA Langley Research Center under Interagency Agreement Number 40-1553-85 with Martin Marietta Energy Systems, Inc.

1. Introduction

The purpose of developing a telerobotic work package for space application is to increase astronaut and overall system safety, productivity, and flexibility. Astronaut safety is of increasing concern because of the number of potentially hazardous tasks, such as hydrazine fuel transfer, being planned for space execution. Astronaut risks increase as the demand for extra vehicular activity (EVA) time increases for work on large projects such as space station assembly, operation, and maintenance activities. A remote system would allow around-the-clock operation while the astronaut-operators remain safely inside the orbiter or space station. Finally, with a telerobotic-based dexterous remote-handling system, operations in the far future can be conducted at significant distances (such as geosynchronous orbit) from the orbiter or space station.

Traditionally, teleoperated manipulators have been designed primarily to operate with low friction and inertia to minimize operator fatigue and backlash and stiffness were of secondary concern. Robots, on the other hand, are designed with high stiffness and minimum backlash as a primary concern to accommodate accurate and rapid positioning; friction and inertia are addressed secondarily, if at all. The design objectives of teleoperators and robots dictate mechanical approaches that are almost mutually exclusive. Attempts to merge these technologies into a "telerobot" have been strictly limited by these contradictory approaches. To accomplish this merger, a joint mechanism is needed that provides very low friction and inertia to accommodate teleoperator requirements and high stiffness and zero backlash to accommodate robotic requirements. A joint mechanism has yet to be developed that can optimize both of these requirements. However, a joint mechanism that approaches this optimal performance has been developed for NASA Langley, Automation Technology Branch. It consists of a traction drive differential that uses variable-loading mechanisms (VLM) and is called the Laboratory Telerobotic Manipulator (LTM).

2. Traction-Drive Joint Mechanism for the LTM

The LTM is a seven-degree-of-freedom telerobot that employs replicated traction drive joint mechanisms as shoulder, elbow, and wrist joints (Fig. 1). Each joint mechanism provides pitch and yaw motions about orthogonal axes. Each joint is attached to the adjacent joints by means of only four fasteners to produce a modular mounting arrangement that allows the LTM arms to be easily assembled and disassembled. This modularity also allows the LTM arms to be easily reconfigured for changing requirements and permits maintenance on the arms by simple module replacement.

The LTM has load capacities to accommodate man-equivalent operation. Each LTM arm has a peak load capacity of 30 lb and a continuous load capacity of 20 lb. To accomplish this requirement effectively, the LTM arm was configured by joints having different torque capacities. The resulting torque requirement for each joint is 435 in.-lbs for the wrist, 960 in.-lbs for the elbow, and 1650 in.-lbs for the shoulder. To reduce the fabrication and engineering cost, a large joint having a peak torque capacity of 1650 in.-lbs is used at both shoulder and elbow positions. In an effort to optimize dexterity and minimize weight, a small joint having a peak torque capacity of

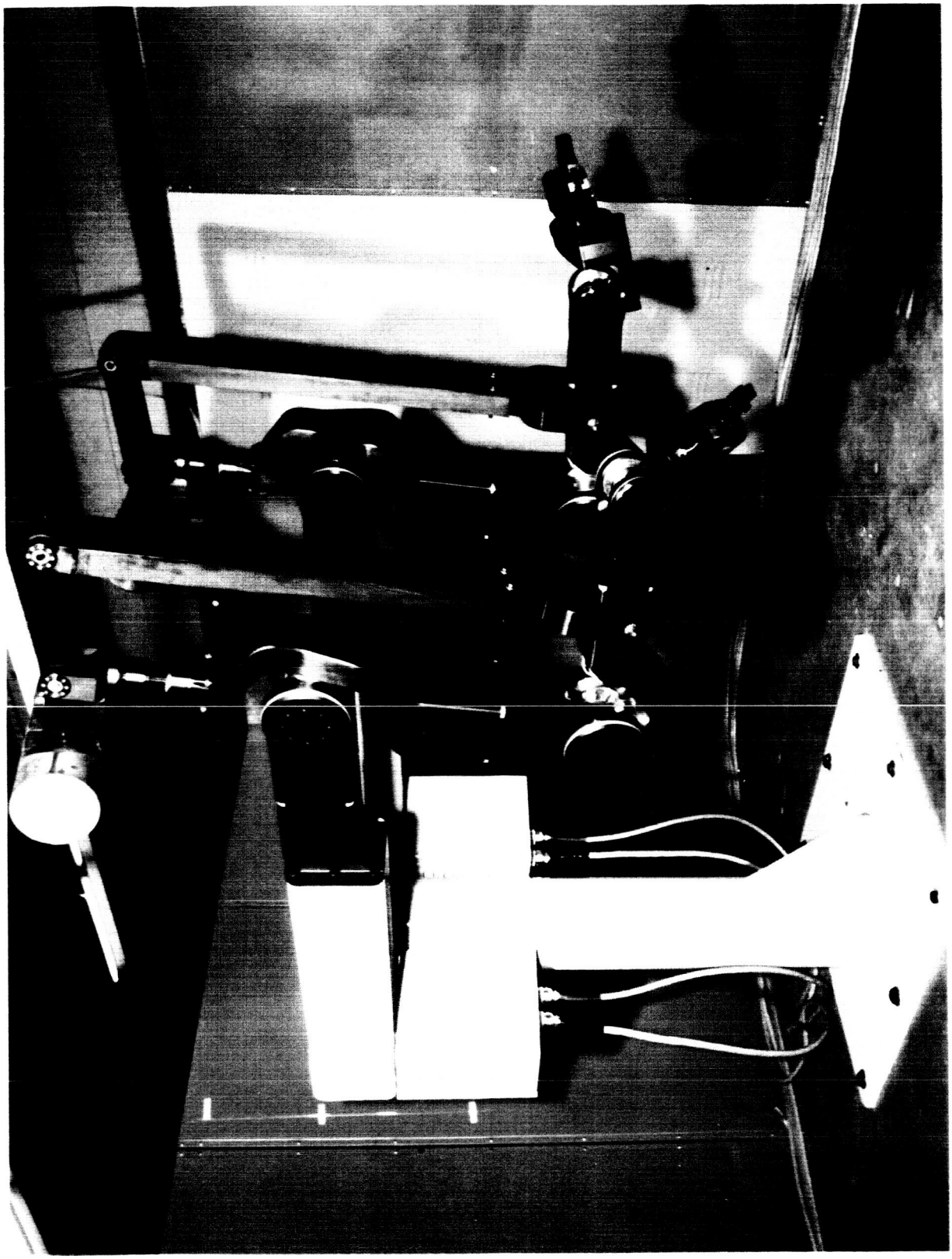


Fig. 1. Laboratory Telerobotic Manipulator (LTM) Slave.

435 in.-lbs is used as the wrist joint. An assembly of the small joint is illustrated in Fig. 2. The large joint is simply an enlarged replica of the small joint and is illustrated in Fig. 3. Both joint assemblies consist of a differential drive mechanism, two DC servomotors (Inertial Motors) with gearheads, two torque sensors, and two resolvers as shown in Figs. 2 and 3. The speed-reduction ratio through the differential is $\sim$3-1/2 to 1. Special gearhead (Bayside Controls) with spring-loaded antibacklash gear trains were used. Commercially available (GSE) torque sensors have been modified and incorporated directly into the joint mechanism to produce a compact arrangement. Vernitron resolvers are located at each joint axis and are coupled directly to the axis of rotation. These resolvers and torque sensors provide the control system data indicating the joint's payload and position.

Cabling provisions have also been made to eliminate the use of external pigtails and connectors. A through-passage within the differential has been provided to accommodate the cabling bundle. This cabling bundle is also equipped with electrical connectors positioned at each mounting interface that engage and disengage automatically as each joint is attached and detached to the adjacent joint.

Permanent-magnet fail-safe brakes have recently become commercially available (Electroid) and have been coaxially mounted to each drive motor. These brakes will safely stop each LTM arm during power failure and will provide the capability of supporting maximum payloads for long periods without motor overheating. The operating principle of a permanent-magnet brake is similar to that of a standard spring-set brake in the sense that permanent magnets are used to generate a magnetic force that replaces the spring force of the spring-set-type brakes. When the coil of a permanent magnet brake is energized, it cancels this magnetic force, releasing the clamping force on the drive disc. The real advantage of these brakes is the amount of torque per unit size and weight. These magnetic units are capable of supplying five times the torque-to-weight ratio as spring-set brakes.

The differential drive mechanism has two inputs and one output which rotate about orthogonal axes. Force transmission through the differential drive mechanism is accomplished by traction drives. Unlike force transfer through gear teeth that generate torsional oscillation as the load transfers between teeth, force transfer through traction is inherently smooth and steady, without backlash, and relatively stiff.[1] The elements of this traction differential drive can be seen in Fig. 4. Two driving rollers provide input into the differential. A significant advantage in this setup is that each driving roller is required to transmit only one-half of the total torque necessary to make a particular motion. These rollers drive two intermediate roller assemblies, which in turn drive the pitch/yaw roller about the pitch and yaw axes. The axis about which the pitch/yaw roller rotates depends on the direction of rotation of the driving rollers. The pitch/yaw roller is driven about the pitch axis when the driving rollers rotate in opposite direction. When both driving rollers are rotated in the same direction, the pitch/yaw roller is driven about the yaw axis. The rolling surfaces of the differential are gold plated in an ion-plating process developed by NASA Lewis Research Center.[2] This plating serves as a dry lubricant in the sense that it prevents the substrates from contacting. Vernitron resolvers are located at each joint axis in an effort to maximize

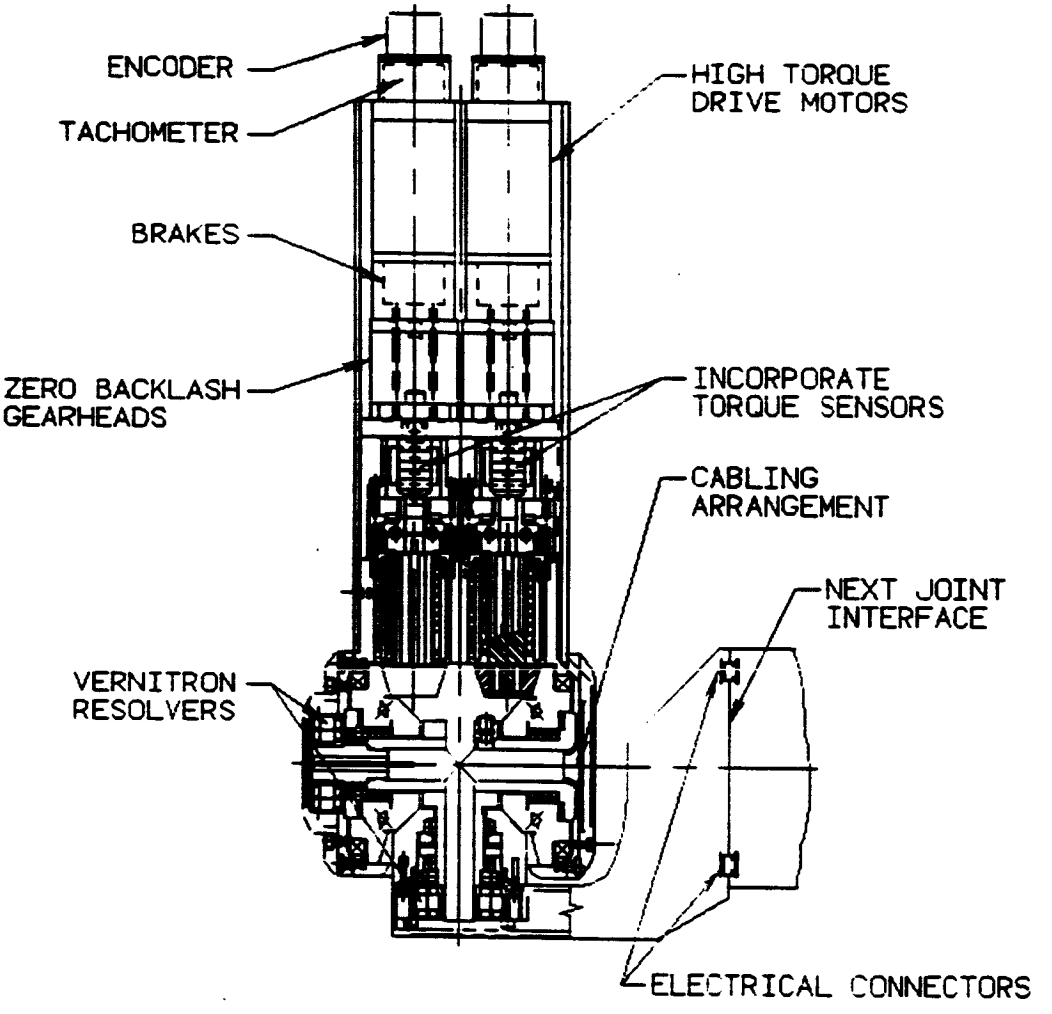


Fig. 2. LTM small pitch/yaw joint assembly.

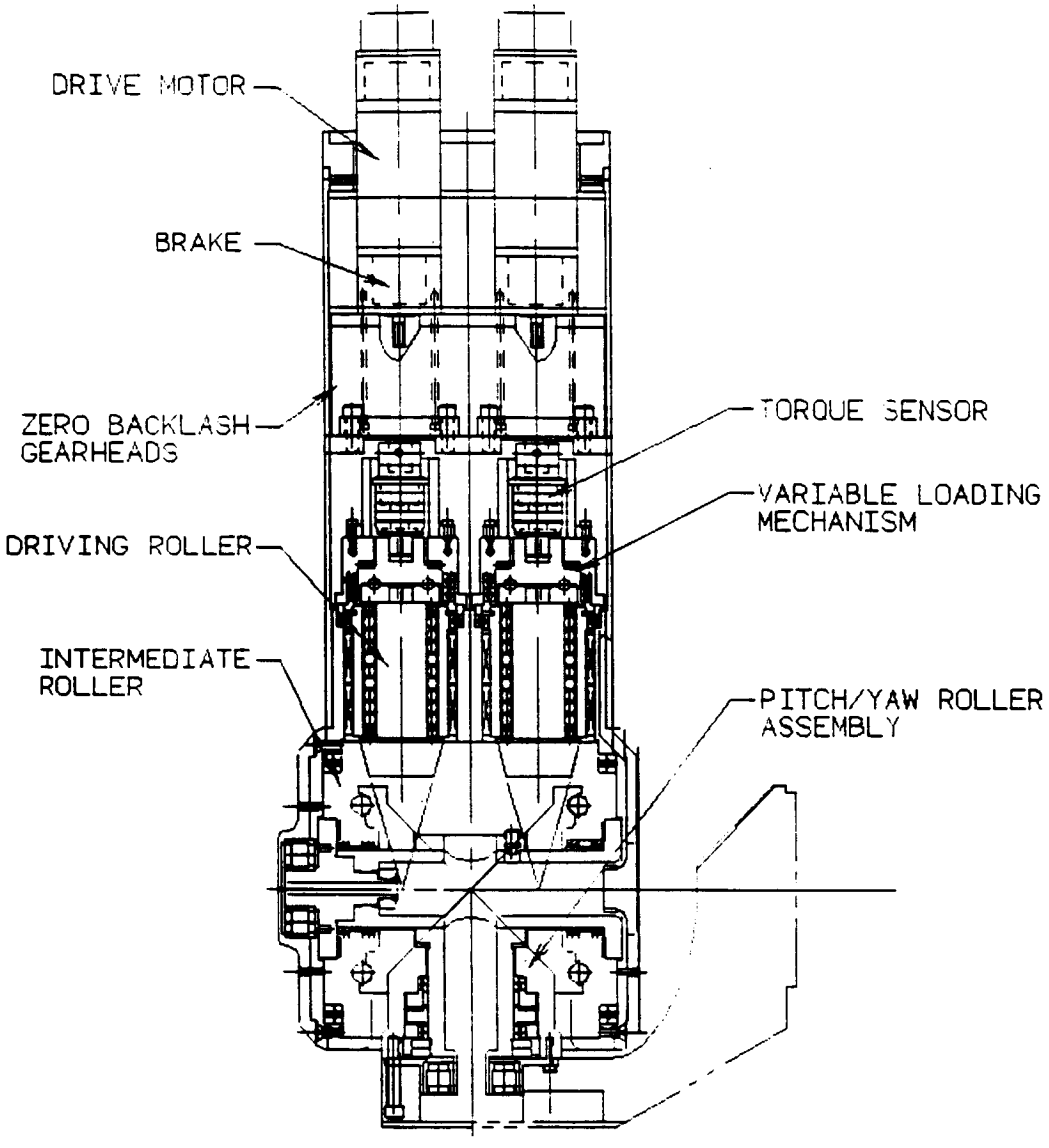


Fig. 3. LTM large pitch/yaw joint assembly.

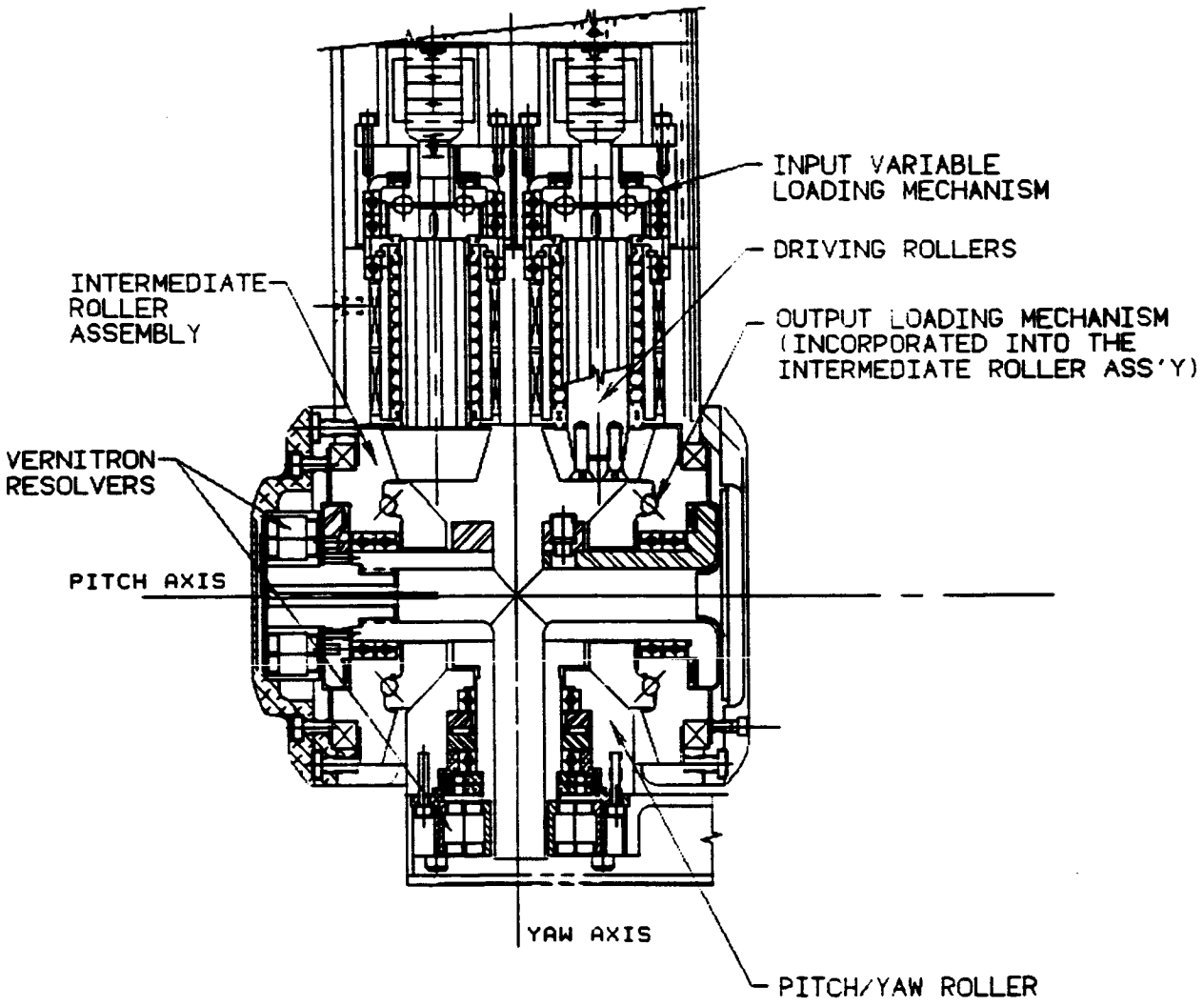


Fig. 4. LTM traction drive differential.

positioning accuracy. By locating these resolvers directly at each joint axis, any creep events that occur through the traction drive differential will not effect the positioning characteristics of the LTM.

VLMs have also been employed as an alternative to constant-loading mechanisms in an effort to improve the differentials back-driveability, mechanical efficiency, and fatigue life. Constant-loading mechanisms produce a constant normal load between the traction drive rollers. This constant normal load must be sized to ensure adequate traction at the joints maximum torque capacity. The obvious disadvantage of this constant normal load is that the traction drive rollers and their supporting bearings are needlessly overloaded during periods of low torque transmission. This constant normal load not only generates extra bearing losses at low torque transmission but, more important, shortens the drive systems fatigue life.[3] To ensure adequate traction with minimum friction loss, VLMs were developed. These mechanisms produce varying normal loads between the traction rollers that are proportional to the transmitted torque.[4] Two VLMs variable loading mechanisms have been incorporated into the traction drive differential. These VLMs are known as the input VLM and the output VLM.

The input VLM produces a varying normal load between the input roller and the intermediate roller assembly. This mechanism consists of a upper thrust cam, a lower thrust cam, a thrust bearing, two radial bearings, a thrust bearing retainer, and four ball bearing balls, referred to as cam balls as shown in Fig. 5. This mechanism generates a thrust force proportional to the input torque. This thrust force is applied to the input roller and is counteracted by the thrust bearing and bearing retainer. The radial bearings provide stability to the upper thrust cam. The upper and lower thrust cams are equipped with tapered contours that are formed by helical grooves. These contours contain cam balls as illustrated in Fig. 6. Each contour is formed by two helical grooves, one cut on a right-hand helix and the other cut on a left-hand helix. These two helical grooves converge at a depth that is slightly less than that of the cam ball radius (0.031 in). A free-body diagram of the upper thrust cam and lower thrust cam is shown in Fig. 7. The input torque (T_i) is transmitted from the upper thrust cam to the lower thrust cam by a compressive force generated in each cam ball. This compressive force F is normal to the tangent helical groove and is the resultant force of a horizontal force F_T and a vertical force F_L. Force F_T is the tangential force required to transmit the input torque T_i. Force F_L is a varying thrust load that is counteracted by the thrust bearing and bearing retainer shown in Fig. 5. This varying thrust load is applied to the input roller and produces a varying normal load between the input roller and intermediate roller assembly.

The output VLM produces a varying normal load between the intermediate roller assembly and the pitch/yaw roller. This mechanism is incorporated into the intermediate roller assembly as shown in Fig. 6. It consists of the intermediate drive roller, eight cam balls, and an intermediate transversing roller. These rollers contain tapered contours that work in conjunction with the cam balls in the same manner as the upper and lower thrust cams of the input VLM. As torque is transmitted between the intermediate drive roller and intermediate transversing roller a thrust force F_L is generated that produces the varying normal force F_N.

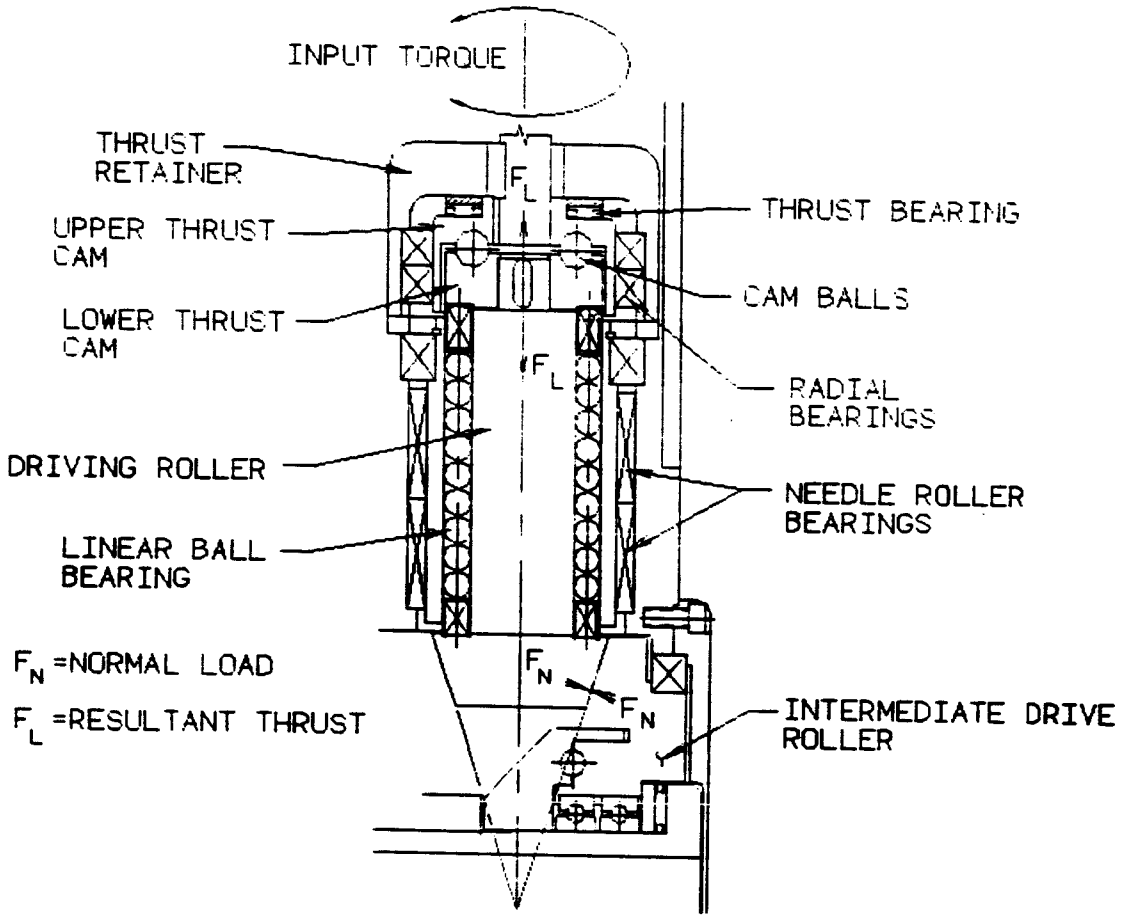


Fig. 5. Input variable-loading mechanism.

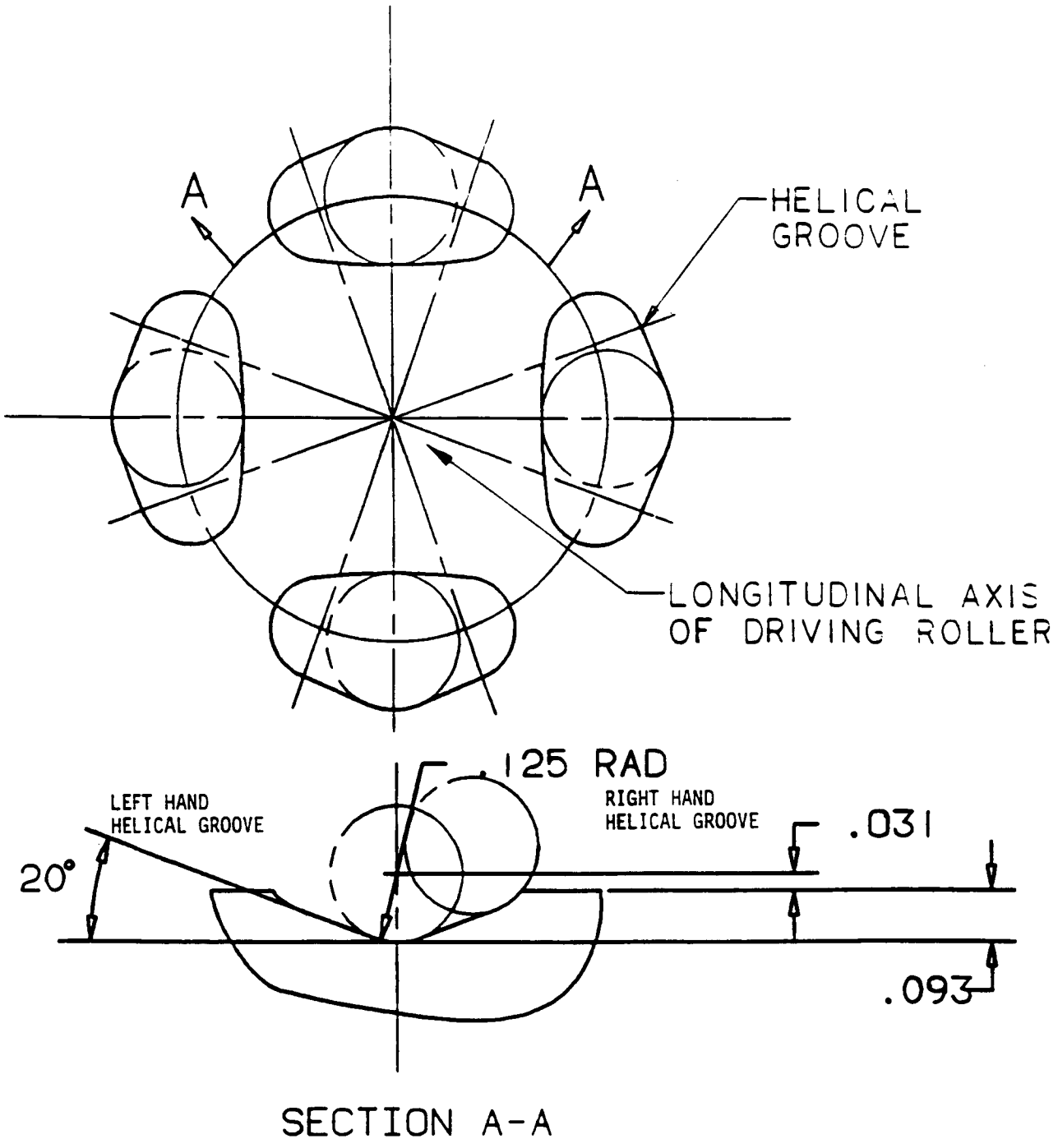


Fig. 6. Thrust cam groove detail.

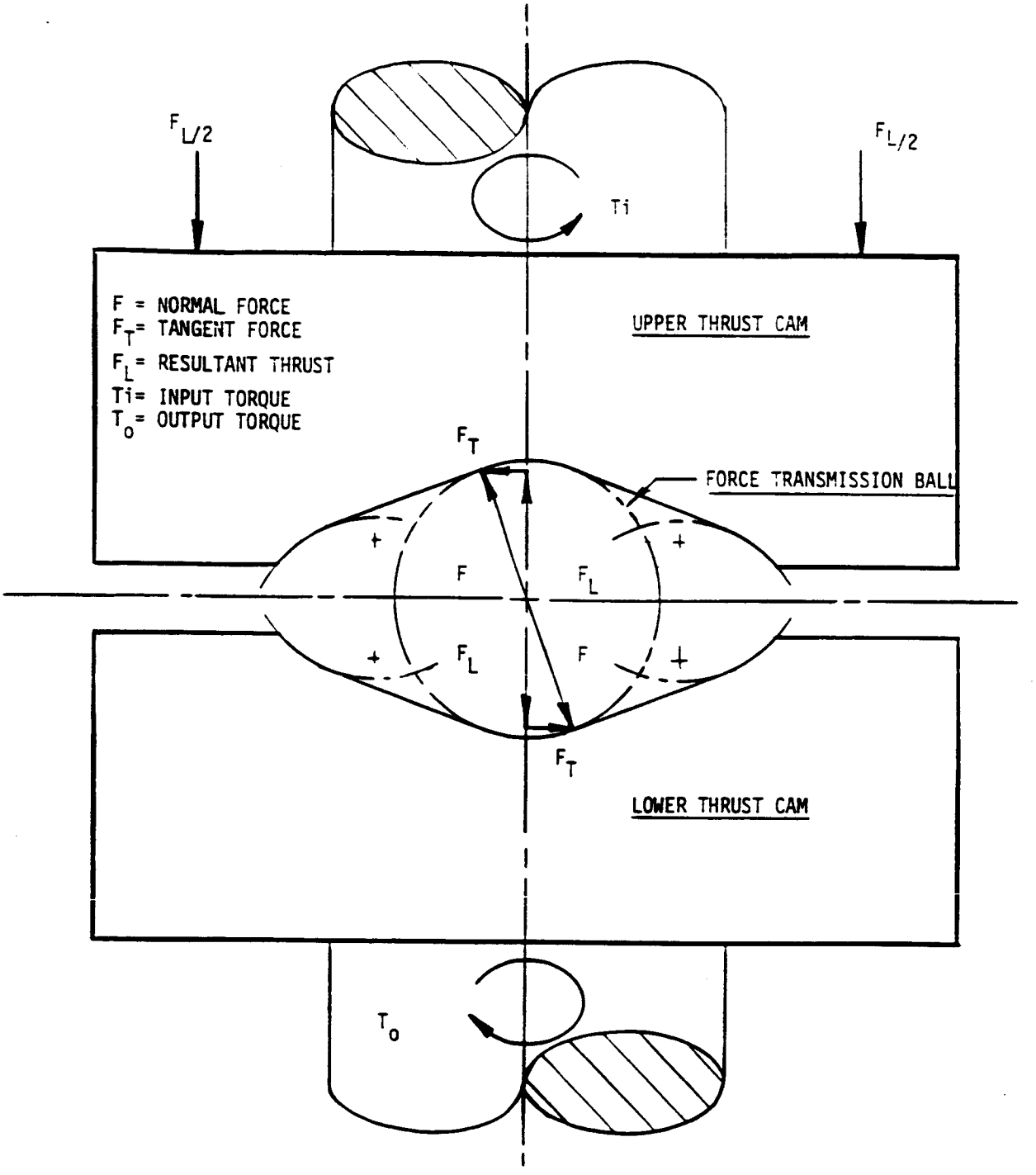


Fig. 7. Free-body diagram of variable loading mechanism.

The operational performance of the LTM was verified through testing during its preliminary design. A photograph of the test stand used is shown by Fig. 8. The test stand was originally designed to accommodate two different types of speed reducers; a power hinge reducer, which was seen to be economically unfeasible; and a harmonic drive reducer, which is now being used. The test-stand differential is very similar to the LTM small-joint differential. Similar bearings and traction drive rollers are employed in both cases. The test stand is equipped with an input VLM and an output constant-loading mechanism. This arrangement provides the capability to compare the two different types of loading devices. Some of the parameters tested were the starting torque, back-driveability, mechanical efficiency, and torque capacity. The test stand demonstrated that a traction drive differential equipped with VLMs will satisfactorily transmit its designed torque capacity with a mechanical efficiency of $\sim$90%. Testing also indicated that a VLM generates only 25% of the starting and back-driving torques, whereas the constant-loading mechanism generated 75% of these differential torques. This appears to indicate that the VLM may reduce the starting and backdriving torque $\sim$50%.

3. Conclusions

A joint mechanism for a space telerobot was developed for NASA Langley Research Center. This joint mechanism incorporates a traction-drive differential that is equipped with variable preload mechanisms. It meets the requirements of both teleoperators and robots. Backlash is eliminated and high stiffness is provided that accommodates accurate and rapid positioning needed in robots; and low friction and inertia is obtained to minimize operator fatigue needed in teleoperated manipulators. By meeting the requirements of teleoperated manipulators and robots, this joint mechanism is the first operational system to mechanically merge these two technologies into a "telerobot".

4. References

1. Stuart H. Loewenthal, Douglas A. Rohn, and Bruce M. Steinetz, "Applications of Traction Drives as Servomechanisms," presented at the 19th Aerospace Mechanism Symposium.

2. William Anderson, Bruce M. Steinetz, and Douglas A. Rohn, "Evaluation of a High-Torque Backlash-Free Roller Actuator," presented at the 20th Aerospace Mechanism Symposium.

3. Rothbart, ed. Mechanical Design and Systems Handbook – 1985, Loewenthal, Zaretsky, Traction Drives, Chap. 34, 1985.

4. Stuart H. Loewenthal and Erwin V. Zaretsky, "Design of Traction Drives," NASA Reference Publication 1154, 1985.

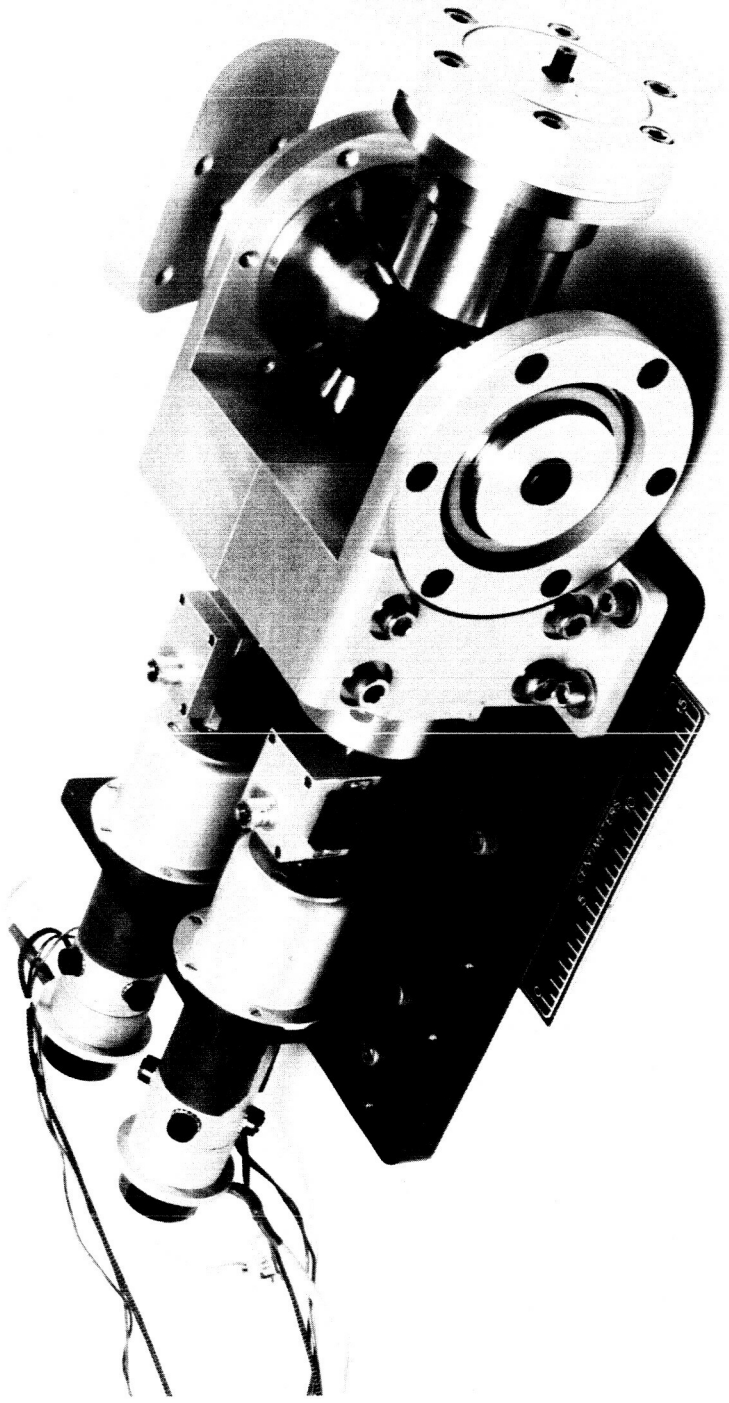


Fig. 8. LTM test stand.

FORCE/TORQUE AND TACTILE SENSORS FOR SENSOR-BASED MANIPULATOR CONTROL

H. Van Brussel, H. Beliën, Bao Chao-Ying

Katholieke Universiteit Leuven
Department of Mechanical Engineering
Celestijnenlaan 300B
B-3030 Leuven, Belgium

Abstract

The autonomy of manipulators, in space as well as in industrial environments can be dramatically enhanced by the use of force/torque and tactile sensors.

In a first part the development and future use of a six-component force/torque sensor for the Hermes Robot Arm (HERA) Basic End-Effector (BEE) is discussed.

Further, a multifunctional gripper system based on tactile sensors is described. The basic transducing element of the sensor is a sheet of pressure-sensitive polymer. Tactile image processing algorithms for slip detection, object position estimation and object recognition are described.

1. Introduction

The HERA is a symmetric six-degrees-of-freedom manipulator arm with an anthropomorphic configuration and an overall length of 11.2 meter. It is designed to perform following operational functions: capture, berthing, release, inspection, insertion and retraction, transfer, placement, actuation, tool operation, EVA-support. It can be operated in the following modes: automatic mode, (tele)-operator-controlled mode, single-joint mode.

Several of the above mentioned functions require the use of closed loop control strategies, based on active force feedback [1]. This requires the presence of a multi-component force/torque sensor imbedded in the HERA BEE (fig.1).

Active force feedback seems to be an appropriate control mode for all "compliant motion" functions. These are functions where the manipulator is in direct contact with its environment (e.g. insertion). Good results have been obtained in industrial environments with force-around-position control loops [2,3]. These schemes also seem applicable to space manipulators [1]. The main difference with respect to industrial manipulators is the back effect of the contact forces on the position loops which has to be taken into account in space manipulators [1], but can be neglected in industrial robots.

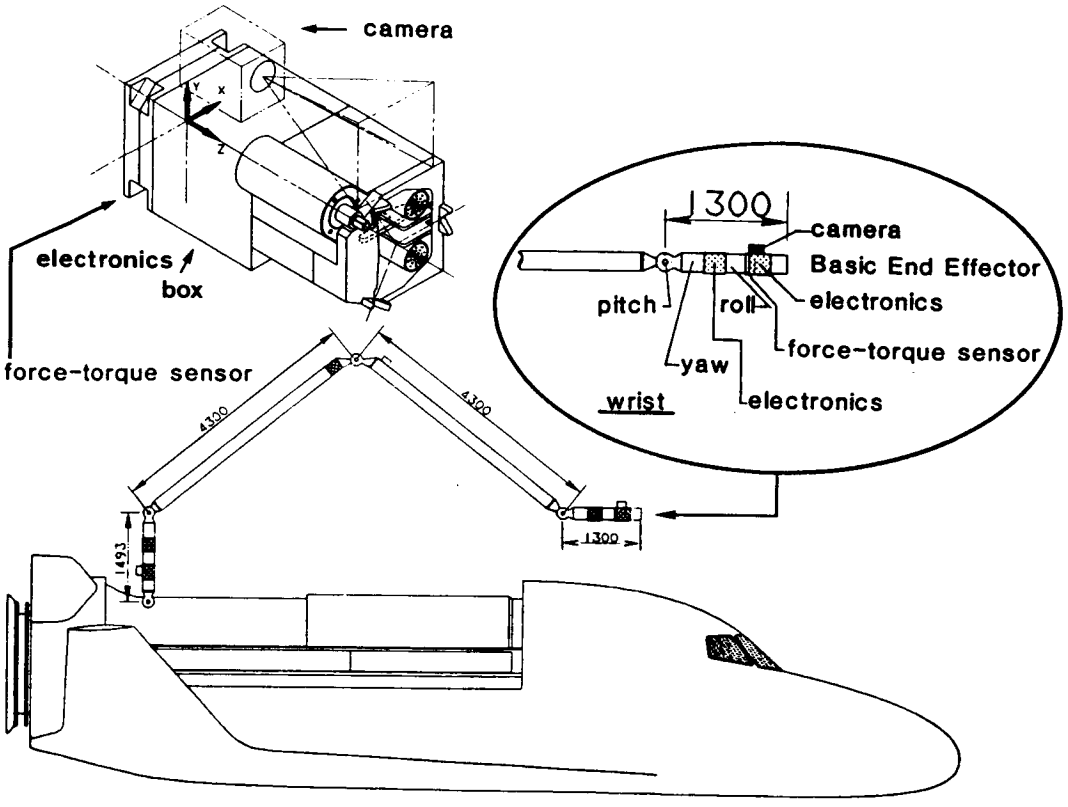


Fig.1. General layout of HERA-arm on HERMES.

At KULeuven, a prototype six-component force/torque sensor for the HERA has been developed and tested. Some particular design features are outlined hereafter.

2. The KULeuven force/torque sensor

An extensive expertise in the development of force/torque sensors for robot (and automotive) applications has been built-up at KULeuven over the last ten years [4]. A CAD-package has been developed for dimensioning these sensors, based on the desired force/torque ranges and the allowable outside dimensions [5]. It takes into account all second order effects and allows dimensioning against fatigue failure.

The most challenging problem associated with the HERA-sensor are the conflicting requirements between the force and torque ranges. Force ranges are 200N versus torque ranges of 150 Nm. These unusually large torques are due to the particular design of the HERA BEE, where a long electronics control box is positioned between the sensor and the tool flange (fig.1). Another challenge was connected with the required stiffnesses: an easily obtainable figure of 10^5N/m for translations, but a very high 2.10^5Nm/rad for rotations.

From the outset, one of the most important design criteria was to achieve *mechanical decoupling*. This results in a diagonal calibration matrix and a dramatically reduced data processing effort. Cross sensitivities of a few

percent are permissible and can still allow perfect force control thanks to the compensating actions of the active force feedback loops around the position loops [6] in the manipulator control structure.

The sensor consists of four cantilever beams in a cross-configuration, at one side rigidly connected to a central block (with a central hole for the passage of a wire harness) and at the other side connected to a rigid outside ring through flexures. These flexures provide four degrees of freedom (and thus two restrictions) to the cantilever beams (fig. 2).

This configuration provides a load situation where only a horizontal force F_h and a vertical force F_v act on the "free" ends of the measuring beams. For nominal loads F_x, F_y, F_z, M_x, M_y, M_z the following relations are valid (neglecting the flexure stiffnesses) on each beam:

$$F_{hx} = \frac{F_x}{2} + \frac{M_z}{2L_t} \quad ; \quad F_{hy} = \frac{F_y}{2} + \frac{M_z}{2L_t} \tag{1}$$

$$F_{vx} = \frac{F_z}{4} + \frac{M_x}{L_t} \quad ; \quad F_{vy} = \frac{F_z}{4} + \frac{M_y}{L_t} \tag{2}$$

The associated strains can readily be calculated. At this stage, design parameter L_t partially determines the relative sensitivity with respect to different load components. In the present case, with high torques at low force levels, L_t is however restricted due to other reasons (e.g. max. outside dimensions). Therefore, in the present design the measuring beams have been laid out diagonally with respect to the square outside shape (fig.2).

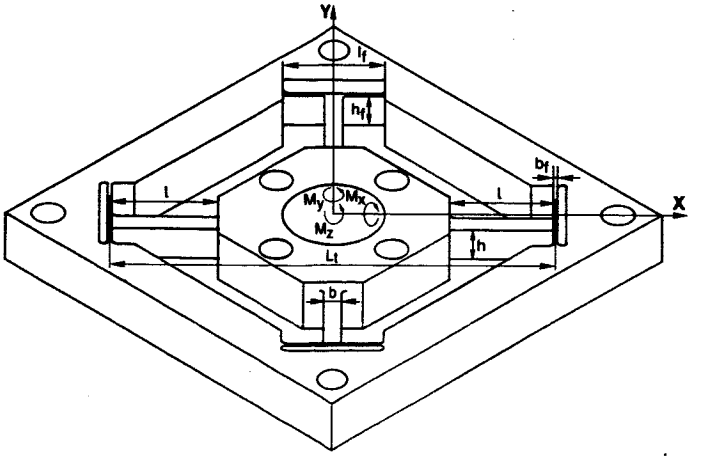


Fig.2. Mechanical layout of HERA BEE six component force/torque sensor.

Having selected L_t, measuring beam cross section dimensions b and h can be calculated in such a way that a maximal strain level ($\epsilon = 0.002$ for the used material Al7075T6) is not exceeded. A ratio $\epsilon_{Fv}/\epsilon_{Fh} = 1$ is adopted as design criterion. Herewith the complete relation between the generated strains for each force/torque component is fixed, so that the length l of the measuring beams can be determined. In chosing this length the sensor stiffness can be controlled.

For the above mentioned specifications, this results in following dimensions: b = 10mm, h = 16.9mm, l = 45mm, L_t = 200mm, resulting in following strain levels: $\epsilon_{Fx,Fy} = 221$, $\epsilon_{Fz} = 66$, $\epsilon_{Mx,My} = 976$, $\epsilon_{Mz} = 821$.
These values are absolute maxima, obtained under the assumption of ideal 4 d.o.f. flexures. At this stage, a more detailed calculation is made, taking into account the influence of the non-ideal flexures. This influence has to be minimized by chosing proper flexure dimensions. In our case the flexure dimensions are: l_f = 40mm, b_f = 1.3mm, h_f = 16.9mm. With this complete load situation, the ultimate stresses and stiffnesses are calculated (Table 1).

Table 1. Design data for the HERA force/torque sensor

Direction	Nominal strain ϵ (microstrain)	Stiffness (N/m,Nm/rad)
x,y	179	6.7×10^6
z	65	38.1×10^6
x,y (rot)	954	1.83×10^6
z (rot)	810	1.37×10^6

As can be seen there, the translational stiffnesses exceed largely the specified ones, while the rotational stiffnesses are slightly below the specifications.

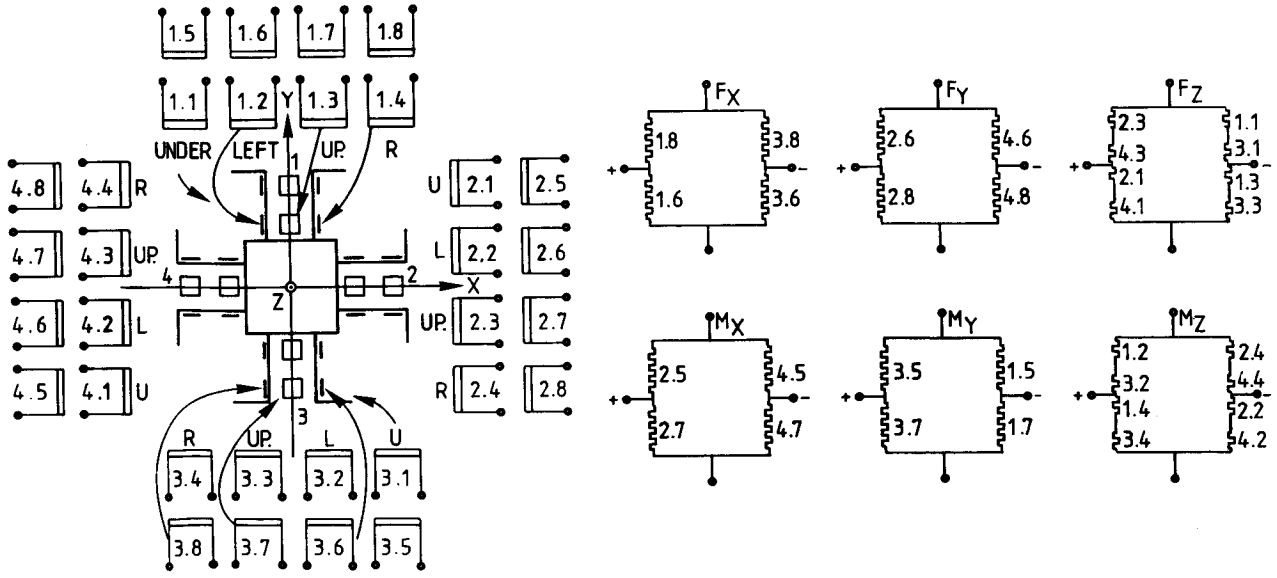


Fig.3. Placement of strain gauges for achieving a decoupled sensor coupling matrix.

A mechanical overload protection is provided by means of mechanical stops, preventing further deformation when 125% of the nominal load is exceeded.

Six strain gauge bridges, positioned at strategic places, to obtain full decoupling of the different force/torque components, are used. Full bridges with four strain gauges are used for F_x, F_y, M_x and M_y, and with eight gauges for F_z and M_z, resulting in a total number of 32 gauges for the complete sensor (fig. 3). Precision instrumentation amplifiers (type AD5245) guarantee good performance over the extended temperature range of -55°C to 125°C.

A static calibration was performed to obtain the 6x6 sensor coupling matrix **A**, relating the force vector **F** to the sensor output signal vector **S**, as follows:

$$\mathbf{S} = \mathbf{A}.\mathbf{F} \tag{3}$$

By applying one force component and measuring the different bridge outputs, the corresponding column of **A** can be determined.

As the sensor is decoupled, only the diagonal elements are significant. Due to the fact that **A** is square and approximately diagonal, inversion is also straightforward:

$$\mathbf{F} = \mathbf{A}^{-1}.\mathbf{S} \tag{4}$$

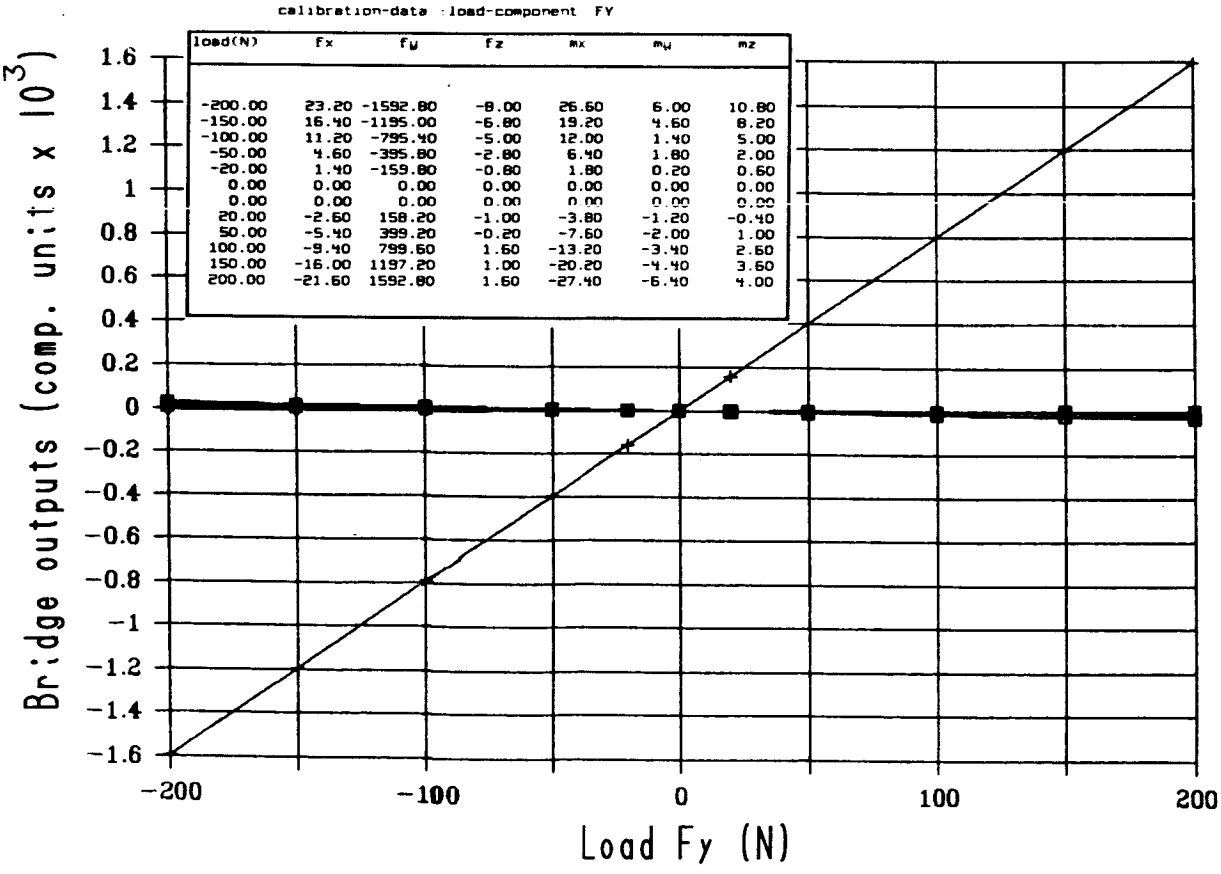


Fig.4. Sensor outputs for pure force load F_y.

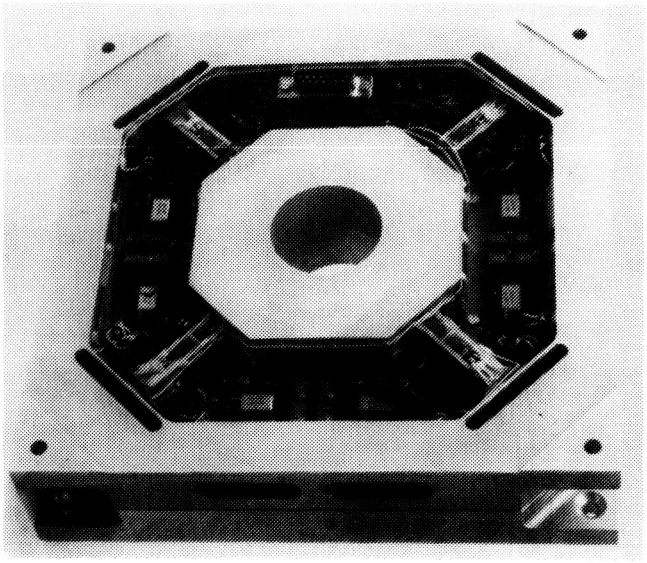


Fig.5. Photograph of finished sensor prototype with built-in electronics.

In case of low cross-sensitivities (≈ 5%) this procedure is acceptable for using the sensor in closed-loop applications of active force feedback [4]. Otherwise, the complete matrix has to be inverted. Fig. 4 indicates a typical calibration result indicating the different bridge outputs for a force F_y applied to the sensor over its full range (± 200N). As can be seen, the cross sensitivity is very low. For the whole sensor, the max. cross sensitivity was smaller than 5%. (A few exceptions were due to incorrectly placed strain gauges and an inaccurately machined sensor body).

Fig. 5 shows the finished sensor prototype with its built-in data processing electronics. The mechanical stops are not visible as they are hidden in the back plane.

3. A sensory controlled gripper system

The above described force/torque sensor can enhance the autonomy of manipulators performing compliant motion tasks, by applying active force-feedback. Additionally, the gripping capabilities of those manipulators can be made more intelligent by introducing tactile perception within the gripper.

At KULeuven some years ago, a high-resolution tactile sensor has been developed to be incorporated into a two-jaw gripper mechanism and aimed at slip-detection, object localisation and recognition [7].

The general layout of the sensor is outlined in fig. 6. It consists of a pressure-sensitive contact layer mounted on a specially laid-out printed circuit board consisting of row and column tracks. In this way the sensor surface is divided into a matrix of 16x16 "islands" (cells). By using a scanning mechanism, these islands are consecutively electrically isolated from their neighbours, thus allowing to measure the local electrical resistance. This latter is function of the pressure exerted on that cell. The digitisation module provides either binary pressure information per cell (through a comparator) or real digital information according to the analog output (through an A/D-convertor).

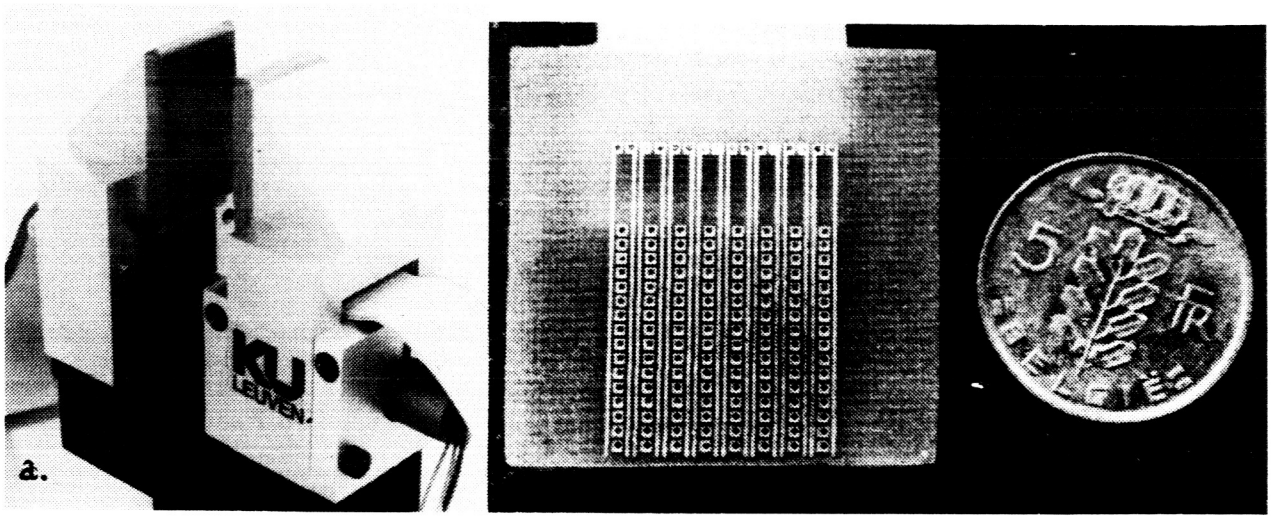


Fig.6. Layout of the tactile sensor built into a two-jax gripper (a); detail of scanning pcb (pressure sensitive layer removed).

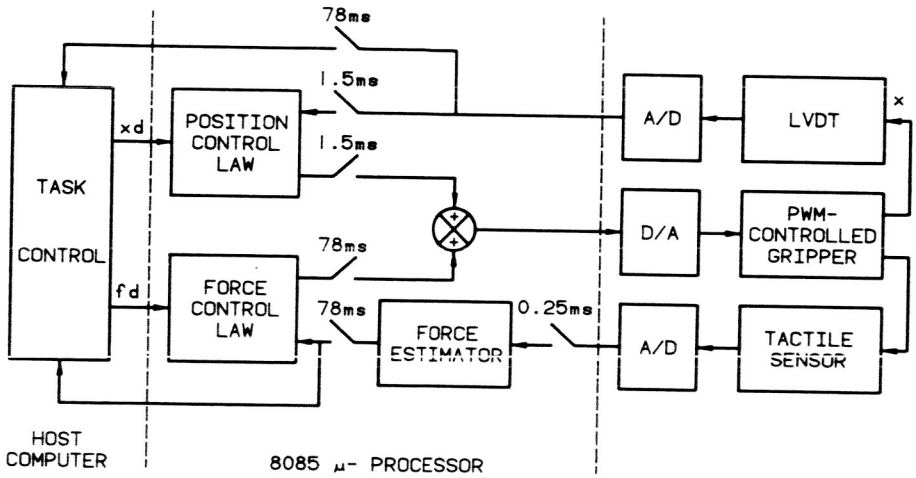


Fig.7. Block diagram of the gripper controller.

The main features of this sensor include:
- matrix size: 16x16 cells (or 32x32);
- spatial resolution: 1.2 mm (or 0.6mm);
- allows detection of 256 pressure levels from 1N/cm^2 to 50N/cm^2 per cell. (The uncertainty level is 4 bits, leaving a real resolution of 16 distinct levels);
- a total acquisition time for 2 x 256 cells of 75ms;
- a wide operating temperature from -30°C to 100°C.

A detailed description can be found in [7]. A further development tested out recently is the implementation of the sensitive layer on an elastic printed circuit board, an interesting feature when one wants to use the sensor on curved surfaces, like the fingers of a dextrous hand.

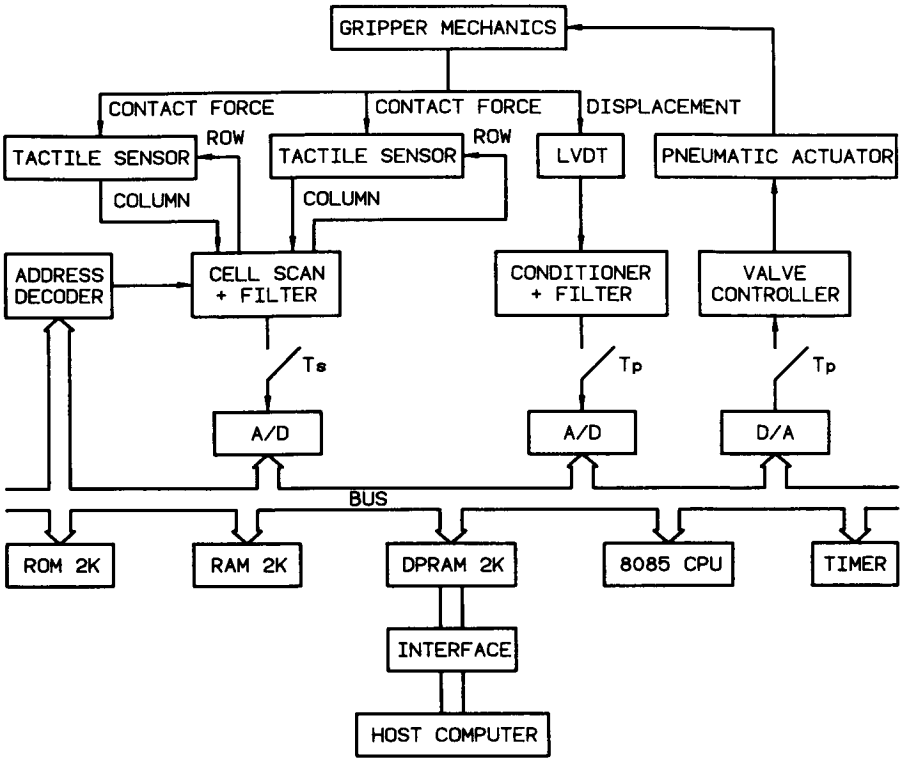


Fig.8. Overall architecture of the sensory controlled gripper system.

Two sensors have been incorporated at the inside surfaces of an off-the-shelf two jaw gripper (fig.6). The global control scheme is illustrated in fig.7. It consists of a hybrid position/force controller. As long as there is no contact force, the system acts as a pure position controlled gripper. Control of the gripper fingers is achieved by a pneumatic piston, driven by a pulse width modulated pneumatic controller based on fast-acting pneumatic valves and pressure transducers. An LVDT displacement transducer provides the position feedback. The tactile sensor acts as force transducer. The overall architecture of the sensory controlled gripper system is illustrated in fig.8.

Slip detection

Slip detection is important in grasping unknown fragile objects. The here described sensors can only detect normal contact forces and no tangential forces. The key point is how to detect slip by only measuring normal forces. Detecting a shift of the gravity center of the tactile image can only work when the sensor's active contact surface is not fully covered by the object. Moreover, image noise normally prevents detection of minute changes in the computed center of gravity. Therefore, the solution adopted here detects *changes in the contact area*. For instance, the contact area reduces and the pressure value of the most loaded cell changes when slip occurs. By combining both features after proper weighing and by using a simple digital filter a very sensitive slip detection method could be worked out. The most noticeable advantage of this solution is that there is no limitation on the size of the grasped object. Experiments have shown very satisfactory results.

Object location estimation

Compared to vision, the use of tactile sensing for identifying object location is advantageous, because:
- much less data is required, reducing image processing time;
- the measurements are direct, without distorsion, shadows, projection errors, etc.;
- no problems occur with obscured objects;
- it is much cheaper than vision;
- location and even recognition are combined with the grasping function.

Important, prior to determining position and orientation of a grasped object, is to start from noise-free tactile images. This is obtained here by dynamic image comparison and proper tresholding. Fig.9 shows the images before and after such filtering. The object's position and orientation coordinates are determined by calculating its center of gravity and the direction of its principal axes of inertia of the enhanced tactile image:

$$x_c = \Sigma x_i/A \quad ; \quad y_c = \Sigma y_i/A \tag{5}$$

$$\theta = 0.5 \arctan \frac{-2\,\Sigma (x_i-x_c)(y_i-y_c)}{\Sigma (y_i-y_c)^2 - \Sigma (x_i-x_c)^2} \tag{6}$$

where x_i and y_i are resp. the column and row coordinates of an active cell i;
A is the number of active cells
x_c, y_c is the location of the center of gravity
θ is the angle between the minor principal axis of inertia and the x-axis (fig.9).

Table 2. Position data and standard directions for cases a, b and c of fig.10.

	$(x_c, y_c) = (\bar{x}_c, \bar{y}_c) \pm (\sigma_{xc}, \sigma_{yc})$	$\theta = \bar{\theta} \pm \sigma\theta$
a	(8.50, 7.50) $\pm$ (0.12, 0.07)	0.0° $\pm$ 0.5°
b	(7.93, 8.04) $\pm$ (0.03, 0.2)	89.94° $\pm$ 0.4°
c	(8.16, 7.84) $\pm$ (0.2, 0.2)	44.81° $\pm$ 0.6°

As an example, a cylinder was grasped 20 times under three typical orientations with respect to the x-axis: 0°, 45°, 90°. Fig.10 shows typical tactile images; table 2 shows the relevant position data and the standard deviations, based on 20 measurements. It can be concluded that the obtained results are very reliable: standard deviations on position coordinates of less than 0.5mm and of 0.6° on the angles, this being obtained with a sensor spatial resolution of only 1.2mm !

Object recognition

An immediate further use of tactile sensors is extracting knowledge from the tactile image to define shape features of the grasped object and making decisions about the class the object belongs to, out of a finite number of

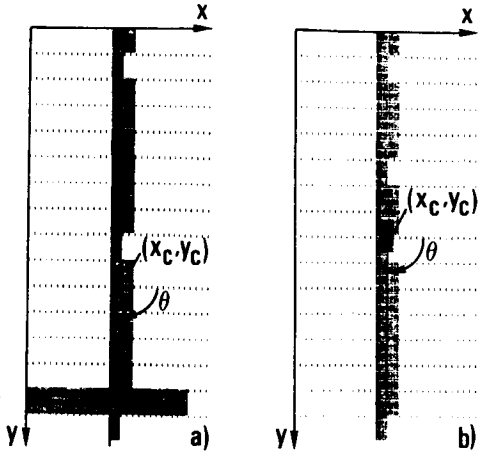


Fig.9. Tactile image before (a) and after (b) noise elimination.

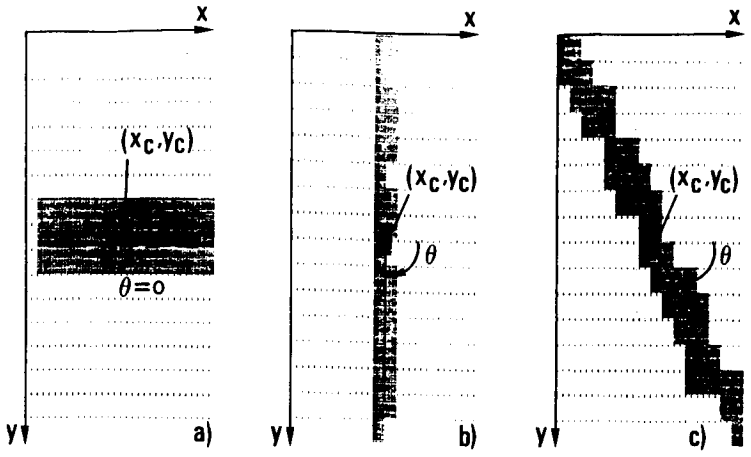


Fig.10. Typical tactile image obtained by grasping a cylindrical object under 0° (a), 90° (b) and 45° (c).

classes. Our tactile sensor can extract following features:
- contact force;
- contact area A;
- perimeter of contact area P;
- moment invariants of A;
- smoothness of contact area, defined by A/P;
- softness of an object.

A recognition programme was developed using the first four features mentioned above, together with the object thickness (measured by the LVDT).

Assume there are n classes of objects and each class has m features f_{ij} (j-th feature of i-th class). An nxm feature matrix **F** can be defined. In this matrix, row i contains the different features for object i, column j contains feature j for all the object classes. During a *learning phase*, with k sample measurements for each object class, the nxm expected value matrix $\bar{\mathbf{F}}$ and the nxm standard deviation matrix $\Sigma = [\sigma_{ij}]$ can be derived using standard statistical techniques. For the *recognition phase*, a 1xm feature row vector $\underline{M} = [m_j]$ is defined for an object to be recognized. Then we compare the matrix $\underline{DF}$ defined by:

$$\underline{DF} = [df_{ij}] \triangleq \bar{F} - [1....1]^T_{1xn}\,\underline{M} = [\bar{f}_{ij} - m_j] \qquad (7)$$

with the standard deviation matrix Σ, element per element. This results in a matrix **P**, with elements p_{ij} defined as follows:

$$p_{ij} = \begin{cases} 0 \text{ if } df_{ij} > 3\sigma_{ij} \\ 1 \text{ otherwise} \end{cases} \qquad (8)$$

Statistically, $p_{ij} = 0$ means that the probability that the j-th object feature belongs to object class i is less than 0.003. Contrarily, $p_{ij} = 1$ means that a large probability exists that the j-th object feature belongs to object class i.

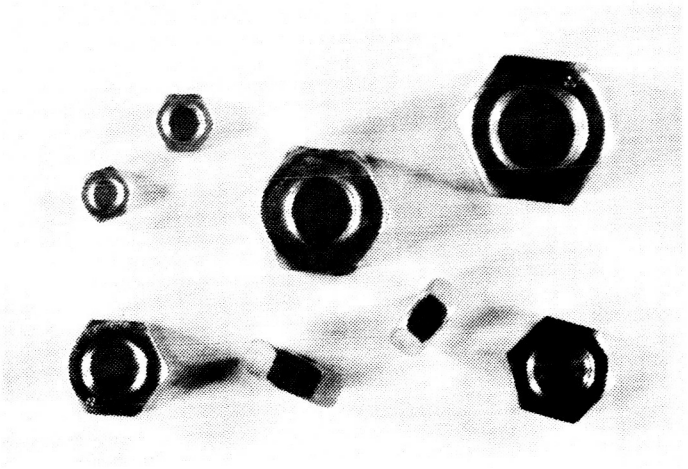


Fig.11. Set of nuts to be recognized by tactile gripper.

The recognition is finally made by means of a score vector **S**, defined by:

$$\mathbf{S} = [s_i]_{nx1} \triangleq \mathbf{P.W} \qquad (9)$$

where $\underline{W} = [w_j]_{mx1}$ is a weight vector, representing the relative importance of the different object features. **W** is determined according to the knowledge obtained in the learning phase.
The object is said to belong to that object class i which yields the largest score S_m, thereby exceeding a certain recognition treshold T_r:

$$S_m = \max \{S_i \geq T_r\},\ 1 \leq i \leq n \qquad (10)$$

This T_r can be determined by trial and error. When no S_i exceeds T_r, then the object does not belong to any class and cannot be recognized.

An experiment was set up to evaluate the performance of the above algorithm. Eight different classes of nuts (see fig.11) were to be recognised by grasping them with the sensory based gripper. Per class, 50 experiments were performed. This resulted in a 100% recognition, without a single failure.

Some observations are appropriate here. First, getting an exact image of an object is not so important for object recognition as for object location. A small image distortion does not influence the recognition result very much. Second, features obtained from other sensors may significantly facilitate recognition, e.g. object thickness information from the LVDT.

4. References

[1] DE SCHUTTER J., 1984, Design techniques for robots, Internal report, KULeuven (Belgium), Dept. of Mech. Eng.
[2] DE SCHUTTER J., VAN BRUSSEL H., 1988, Compliant Robot Motion, I. A Formalism for Specifying Compliant Motion Tasks, Int. J. of Rob. Res., Vol.7, No.4, pp.3-17.

[3] DE SCHUTTER J., VAN BRUSSEL H., 1988, Compliant Robot Motion, II. A Control Approach Based on External Control Loops, Int. J. of Rob. Res., Vol.7, No.4, pp.18-33.
[4] VAN BRUSSEL H., BELIEN H., THIELEMANS H., 1985, Force sensing for advanced robot control. In Proc. 5th Int. Conf. on Robot Vision and Sensory Control, ed. N.J. Zimmerman, Bedford (U.K.): IFS.
[5] NEYRINCK F., 1988, Computer-aided design of force sensors (in Dutch), Thesis 88E1, KULeuven (Belgium), Dept. of Mech. Eng.
[6] DE SCHUTTER J., 1986, Compliant robot motion: task formulation and control, Ph.D.Thesis, KULeuven (Belgium), Dept. of Mech. Eng.
[7] VAN BRUSSEL H., BELIEN H., 1986, A high-resolution tactile sensor for part recognition, In Proc. 6th Int. Conf. on Robot Vision and Sensory Control, ed. M. Briot, Bedford (U.K.): IFS.

REDUNDANT SENSORIZED ARM+HAND SYSTEM FOR SPACE TELEROBOTIZED MANIPULATION

prof.Alberto Rovetta, eng. Paolo Cavestro

Department of Mechanics
Politecnico di Milano,Italy

Abstract

This paper deals with an integrated system, composed of an arm, a wrist, a mechanical multifingered hand, which has been realized in separate parts, and which is on development for possible application in telemanipulation.

The redundancy of the degrees of freedom of the system and of the sensors, the application of logical rules and the supervision of teleoperators may be applied in order to have an optimum of reliability of the system in space telemanipulations.

1.INTRODUCTION

This paper deals with the realization of a mechanical system, (indicated FCR8) for telemanipulation in the space (Fig.1).

Objects of generic shape, of relatively small and big sizes, of different masses and deformabilities, may be manipulated and grasped by a robotized system, which is constituted by an arm (fig.2), a wrist and a multifingered hand (fig.3).

By means of a strict integration among the sensors data , the mechanical and electronic software for the control of hand and arm motion, is possible to perform a multipurpose task in the space.

The redundancy of the system may be utilized to different goals, with specific reference to the telemanipulation.

The recognition in the space application is performed by the hand itself, which touches the object and determines parameters for the optimization of the prehension configuration.

After the recognition phase, the hand is moved to the final manipulation, also with the support of the teleoperator.

A parallel process of simulation and modelization, also with the aid of expert systems, may be performed.

The feasibility project is reported, for the first laboratory prototype.

2.USE OF BASIC PRINCIPLES OF TELEOPERATIONS IN THE DESIGN OF THE MANIPULATION SYSTEM

The basic principles of teleoperations in space, adopted in the development of the feasibility project of the mechanical system FCR8, are reported.

They are:

- internal functionality of the hand, with its own capabilities in the prehension of objects, according to physical principles;
- redundancy of informations and data from the sensors and communications channels from the hand, from the arm and from the environment towards the operator, with the evaluation in every moment of the reliability of every signal and data;
- teleoperations in macrooperations and in microoperations, with different ranges of tolerances and precision, with the supervision of the whole process by a second operator, able to assist the general process, instead of the single phases, and in every case to assist the main operator.

The concept of a double teleoperation is devoted to the redundancy of control, in order to obtain a local intelligent control by one operator, and a whole supervision, during the process, and in the same time with the aims of serving an emergency recovery, in front of unforeseen events.

Reliability in teleoperations and telemanipulations in the space is the main fact in front of the evaluation of the quality of the work in the space.

3.MANIPULATION SYSTEM DESCRIPTION

The manipulation system is composed of a mechanical hand, with three fingers and a palm, (see References /1/), connected to a supporting system, with 6 degrees of freedom.

The hand performs the manipulation process, grasping the elements and moving the single fingers in order to perform the task.

The mechanical system, which supports the hand, presents six degree of freedom in the motion.

It is similar to a robot, where the arm presents 4 degrees of freedom, and the extremity part, analogue to a wrist, presents two degrees of freedom.

The total system presents six degrees of freedom in the support structure and 13 degrees of freedom in the hand.

The strategy of the control system is that the redundancy of the degrees of freedom is fundamental to obtain a manipulation process which offers reliability.

4.DESIGN OF THE SUBSYSTEMS

The design of the hand is obtained with reference to some

basic principles, connected with mechanical and systems laws.

Some of these principles are:

use of three fingers and one palm, to obtain a multiple points contact, to ensure a reliable prehension; use of friction surfaces for the fingers and of smooth surfaces for the palm, to increase prehension stability; use of a settlement phase, able to obtain a sequence of micromotions of the object in the hand, which perform the process of selfadjustment of the object inside the hand; integration of sensors data, by using the redundancy of the sensors output signals for improving the reliability and capability of the control system; development of different parallel sensors, for the control of the prehension process itself.

The design of the arm which supports the hand is developed in order to offer to the hand a support of 6 degrees of freedom, which may be controlled in a parallel way.

5.CHARACTERS OF THE TELEROBOTIZATION

The manipulation process may be subdivided in different phases, both in time and in space.

The first phase is the approach phase, where the manipulation system is moved in order to approach the objects to be manipulated, inside a certain operative radius in the working area.

The mechanical arm and the wrist must be moved in order to get a position, with a large capability of motion and operation.

The control of the motion is obtained according to the rules of the expert systems, in order to have a collision avoidance and a optimal trajectory choice, also considering the reliability of the system.

(The methodology of the expert system application for collision avoidance and optimal trajectory choice has been presented and developed in Ref. /2/.)

After the approach phase, the manipulation phase may begin.

It is subdivided in two parts: the first part is the operation of grasping of the object or of the tool, the second part is the motion of the object or of the tool.

The presence of a double arm may simplifie the procedure, in front of a better dexterity of the system.

In conclusion, the operations are a sequence of macrooperations and of microoperations, which are performed by the system.

The teleoperation is obtained by means of the choice of the strategies of approaches and of the tasks.

The tasks are subdivided in elementary tasks, and the elementary tasks, developed by the elements of the system, are supervised by the teleoperator.

Teleoperation in the system here presented consists in the subdivision of tasks in sub-task.

Teleoperator decides the subdivision of tasks, and the system operates freely inside the subtasks, according to simple boundary conditions or more complex expert systems rules.

The here described task has been under development, in different subsystems, in the Laboratory of Robotics, Department of Mechanics, Politecnico di Milano.

The process of interaction must be developed in the future according to the design rules, represented synthetically in Fig.4.

6.REQUIREMENTS FOR MANIPULATION IN THE SPACE

The FCR8 system presents some features for telemanipulation in the space.

They are mainly:

1) capability to approach the operating position, by the use of inverse kinematics, applied to the mechanical arm, with the choice of optimal trajectories, according to prerequired functions;

2) possibility of developing dynamic control with reliability evaluation of different functions, by the consideration of the reliability functions of the mechanical components (see Ref./3/);

3) adaptability of the hand to different functions, with the selfadjustable prehension and grasping phase, both for the single object or for a working tool;

4) sensitivity of the mechanical hand in front of dynamical perturbations, by means of the motion of the fingers, in order to obtain a stable prehension in grasping and a reliable gripper for the use of the tools;

5) reliability of the system , in front of a mechanical and electric design, which need no lubricant or liquid material, and which may be applied to loads of Nw up to hundred of Nw, with a selfadaptability of the prehension system.

The teleoperation may be performed by means of two levels:

- macrooperations
- microoperations.

The macrooperations may be performed with the support of the sensors which are in the system.

The operating sensors are : force sensors and tactile system in the hand for a first recognition of the unknown object to be grasped; vision and recognition system on the arm, able to determine the characters of the environment.

The integration of the sensors with the "experience" of the robotized system and of the teleoperator is a main character of the system on development.

Multiprocessor system and transputer system are on application for adapting the computers capabilities to the telerobotized system requirements.

7.CONCLUSIONS

An integrated system, composed with an arm, a wrist and a hand with redundancies in the mechanical design and in the sensors, may be used for telerobotics in space, where reliability of the system is a basic element.

The application in the space is indicated in front of different subtasks which may be performed by the robot.

Supervision in the steps of the tasks is obtained by the operator, where a second operator may evaluate the reliability of the single subtasks in front of the results.

This paper has been developed with the support of the Italian Education Ministry.

References

/1/ Nato Workshop on "Robots with Redundancy : Design, Sensing and Control", Salo', Italy, June 1988

/2/ A.Rovetta, R.Sala, Optimization of trajectories in assembly with logical support, Tecniche dell'Automazione, june 1988

/3/ A.Rovetta, M. Battaini, Reliability on robots: an applicative study, Meccanica, in print

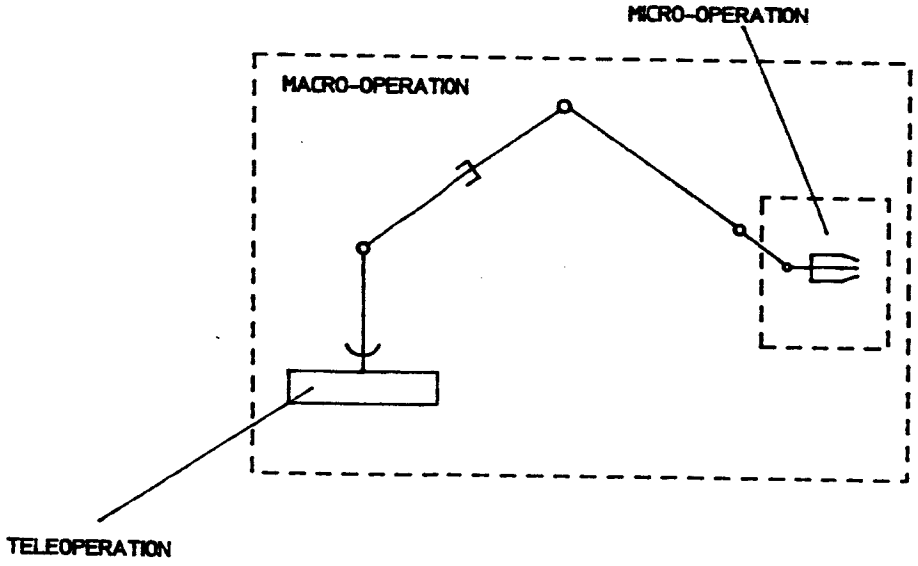


FIG.1 ARM+WRIST+HAND (6+13 DOF)

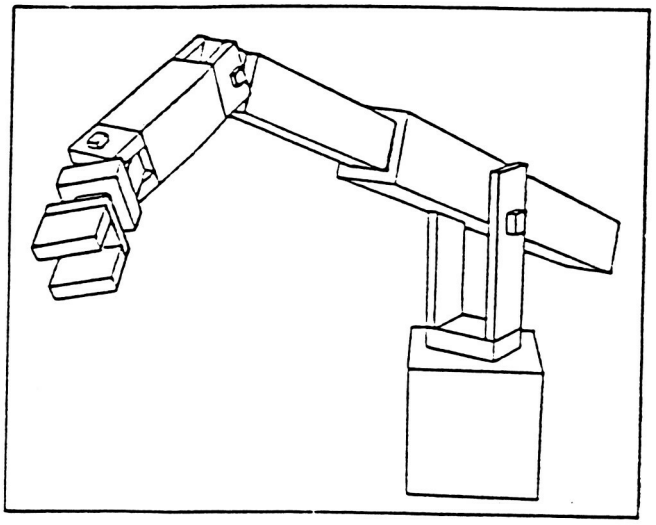


Fig. 2 Mechanical arm with 6 degrees of freedom

Fig. 3 Sensor controlled multifunctional robot hand

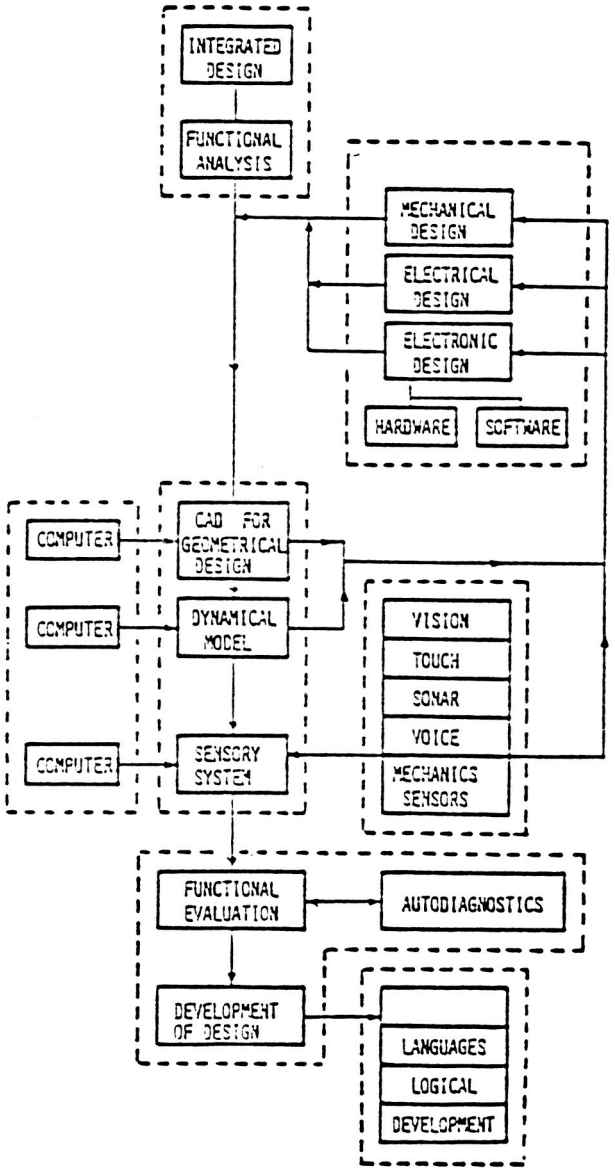


Fig. 4 Conceptual scheme for the system integration

IMPEDANCE HAND CONTROLLERS FOR INCREASING EFFICIENCY IN TELEOPERATIONS

C. Carignan, J. Tarrant

ST Systems Corporation
4400 Forbes Blvd.
Lanham, Md. 20706

Abstract

An impedance hand controller with direct force feedback is examined as an alternative to bilateral force reflection in teleoperations involving force contact. Experimentation revealed an operator preference for direct force feedback which provided a better "feel" of contact with the environment. The advantages of variable arm impedance were also made clear in tracking tests where subjects preferred the larger hand controller inertias made possible by the acceleration feedback loop in the master arm. It is expected that the ability to decouple the hand controller impedance from the slave arm dynamics will be even more significant when the inertial properties of various payloads in the slave arm are considered.

NOMENCLATURE

θ_m = master arm position (rad)
I_m = master arm inertia ($kg.m^2$)
I_{md} = desired master arm inertia ($kg.m^2$)
μ_m = master arm viscosity ($kg.m^2.s^{-1}$)
μ_{md} = desired master arm viscosity ($kg.m^2.s^{-1}$)
K_{ma} = master arm acceleration error gain ($kg.m^2$)
K_{md} = master arm artificial damping ($kg.m^2.s^{-1}$)
K_{mp} = master arm position error gain ($kg.m^2.s^{-2}$)
K_{mv} = master arm velocity error gain ($kg.m^2.s^{-1}$)
K_{mf} = force feedback gain to master arm (.)
u_m = master arm control input ($kg.m^2.s^{-2}$)
θ_s = slave arm position (rad)
I_s = slave arm inertia ($kg.m^2$)
μ_s = slave arm viscosity ($kg.m^2.s^{-1}$)
μ_{sd} = desired slave arm viscosity ($kg.m^2.s^{-1}$)
K_{sd} = slave arm artificial damping ($kg.m^2.s^{-1}$)
K_{sp} = slave arm position error gain ($kg.m^2.s^{-2}$)
K_{sv} = slave arm velocity error gain ($kg.m^2.s^{-1}$)
u_s = slave arm control input ($kg.m^2.s^{-2}$)
F_s = contact force on slave arm ($kg.m^2.s^{-2}$)
τ_h = torque from human operator ($kg.m^2.s^{-2}$)
τ_s = torque input to slave arm ($kg.m^2.s^{-2}$)
θ_e = position of slave arm contact (rad)
K_e = environmental stiffness ($kg.m^2.s^{-2}$)
μ_e = environmental viscosity ($kg.m^2.s^{-1}$)
T = sampling period (s)
ZOH = zero order hold (^=sampled data)
COD = computational delay

1. INTRODUCTION

The work presented here explores two control options for hand controllers used in master/slave teleoperation. The first algorithm, bilateral force reflection (BFR), is the more traditional approach and uses position and velocity error signals created when the slave arm contacts an obstacle to generate a force signal in the master arm (see Fig. 1). This system is force reflective since contact will eventually result in a force signal "reflected" back to the master arm. This algorithm also uses a symmetric control structure by which each arm follows the other using the same error signals. Hence the term "bilateral" is also used in describing this form of control.

The other controller used, impedance control with direct force feedback (IDF), does not exhibit the coupling of the master and slave arms as in bilateral control. The only signal fed back from the slave arm to the master arm is a

strain gauge reading proportional to the force on the slave arm. Hence this system is also force reflective but uses direct force feedback rather than position and velocity errors to generate the signal to the master arm. In principle, the contact force has very little to do with the slave arm dynamics unless the object being pushed on is very compliant. Thus the master arm can be considered decoupled from the slave arm. If the position, velocity, and acceleration of the master arm are measured, then the stiffness, viscosity, and inertia of the master arm can be varied using these signals, respectively. Because the designer has full control over the impedance characteristics of the master arm, this form of control has been named "impedance control."

The main advantage of impedance control over bilateral force reflection is that the "feel" of the master arm can be varied at any time. The master arm can be made to feel "heavier" that the slave arm in proximity operations where sudden movement of the slave arm could be disastrous, or feel "lighter" than the slave arm to reduce operator fatigue when transporting massive objects. Similarly, the viscous effect of friction in the gear box could be eliminated, or viscosity could be added if the natural friction does not provide sufficient damping.

Another advantage in IDF control is the presence of direct force feedback to the operator. Instead of receiving pseudoforce information from the robot, the operator can feel exactly what the actual robot arm feels. Not only can force contact be felt, but inertial forces resulting from the slave arm movement can also be sensed because such motion also results in strain on the arm link. Though force feedback is usually considered superior because of the more exact nature of the feedback, it is less stable than its force-reflective counterpart.

Though impedance control is a relative newcomer to the area of teleoperation, its possible advantages in hand controller applications were seen over two decades ago. Gydikov [1] examined the ergonomic impact of varying the inertia of a rotary hand controller mechanically. Subjects were asked to track light stimuli moving at various speeds, and both hand controller velocity and operator pressure on the handle were recorded. His experiments revealed hand tremors of constant frequency when tracking constant velocity targets. Though his purpose was to seek information for formulating operator models, Gydikov's discovery of inertia-dependent hand tremors provided valuable information regarding an important impedance variable in hand controller design.

More recently, Paines [2] developed and tested an impedance hand controller that translated along a linear carriage. His experiments tested the effect of hand controller inertia and viscosity on the ability of an operator to maintain a constant position during a step input or drive the hand controller to some desired position. Though his main purpose was to study the effect of gravity on operator performance (tests were also conducted in neutral buoyancy), he also discovered hand tremors in the velocity data which depended on the inertia and viscosity of the hand controller.

There are many papers on the use of direct force feedback in robot control applications, but relatively few in which a master arm is used to provide the slave arm commands. Hannaford and Anderson [3] conduct master/slave experiments using a configuration similar to ours with fixed impedances. In addition to the force feedback term to the hand controller, however, they retain the position error term in the master arm feedback which we found to detract from the advantages of direct force feedback. The main purpose of their experiments was to verify the accuracy of the human operator model used in their simulations as well as detect parameters pertinent to hand controller design.

Their experiments revealed significant hand controller chattering on impact when only one or two fingers were used to grab the handle, but reduced significantly when the full hand was used. Perhaps even more significantly, it was found that although adding servo level damping to the hand controller had a similar effect to the additional damping provided by the operator's full hand, the system felt "sluggish" to the operator. This facilitated an argument for a control architecture in which the human operator impedance is continuously measured and only the minimum necessary amount of damping be supplied by the hand controller. It is not difficult to argue a position in favor of adjusting the full hand controller impedance using such monitoring.

As mentioned previously, there are many papers which deal with the issue of force control without the added complication of a human operator in the control loop. One such paper by An and Hollerbach [4] provided the inspiration for our force contact experiments with an eccentric cam. Whitney [5] provides a good overview on force, stiffness, and accomodation control for robot applications. The reader is cautioned that these control terms refer to the inputs to the robot arm and not the type of feedback used, e.g. "force control" means the force is commanded not that the force is fed back. Often, all three sensor readings, position, velocity, and force, are fed back to modify the inputs to the robot arm. The reader may also be interested in referring to an excellent paper by Eppinger and Seering [6] on bandwidth limitations in robot force control when reading the analysis section of this report.

2. DYNAMICS

The block diagrams for the BFR and IDF systems when there is no force contact are shown in Figs. 2a and 2b, respectively. Without force contact, the contact sensor will read only the inertial forces from the slave arm rotation, thus the acceleration feedback term in the IDF case. (The gauge actually reads $-(\rho/L)I_s\ddot{\theta}_s$, where ρ is the master arm's radius of gyration and L is the load calibration position; since most of the inertia originates in the handle, $\rho \approx L$.)

The transfer function between the human input torque to the master arm and the commanded input torque to the slave arm is

$$\frac{\tau_s}{\tau_h} = \frac{H_s G_m}{1+H_m G_m+H_s G_s} \qquad \text{[BFR]} \tag{1a}$$

$$\frac{\tau_s}{\tau_h} = \frac{H_s G_m}{1+(1+K_{mf} I_s G_m s^2) H_s G_s} \qquad \text{[IDF]} \tag{1b}$$

For the continuous-time version, the component transfer functions are as follows:

$$H_m = K_{mv}s + K_{mp} \qquad G_m = \frac{1}{I_m s^2 + \mu_m s} \tag{2}$$

$$H_s = K_{sv}s + K_{sp} \qquad G_s = \frac{1}{I_s s^2 + \mu_s s}$$

If there is no arm damping present, (1) reduces to

$$\frac{\tau_s}{\tau_h} = \frac{K_{sv}s + K_{sp}}{I_m s^2 + (\frac{I_m}{I_s}K_{sv}+K_{mv})s + (\frac{I_m}{I_s}K_{sp}+K_{mp})} \qquad \text{[BFR]} \tag{3a}$$

$$\frac{\tau_s}{\tau_h} = \frac{K_{sv}s + K_{sp}}{I_m s^2 + (\frac{I_m}{I_s}+K_{mf})K_{sv}s + (\frac{I_m}{I_s}+K_{mf})K_{sp}} \qquad \text{[IDF]} \tag{3b}$$

where $I_m=I_s=I$. If the corresponding position and velocity error gains are equal in the BFR case, and K_{mf} is chosen to be unity in the IDF case, then the position and velocity gains for both cases can be determined by

$$K_v = I\zeta\omega_n$$
$$K_p = \frac{1}{2}I\omega_n^2 \tag{4}$$

where ω_n is the desired natural frequency or system "bandwidth" and ζ is the damping ratio.

In the case where the slave arm is in contact with the environment (see Fig. 3), the situation is complicated by the effect of the contact force on the slave arm itself plus the feedback term to the master arm in the IDF case. (Note: we now choose to ignore the inertial force in the sensor dynamics because of the domination of the contact force.) After a considerable amount of block diagram manipulation in which the relay is replaced by unity, one is left with the standard representation shown in Fig. 4 where the plant open-loop transfer function, G, is given by

$$G = \frac{-K_e H_s G_s G_m}{1+(K_e+H_s)G_s} \tag{5}$$

the compensator by

$$H = -K_{mf} + H_m(\frac{1}{K_e} + \frac{G_m}{G}) \tag{6}$$

and the component transfer functions by (2). (Note: for the impedance controller, I_m, μ_m, and μ_s are replaced by the

desired quantities, I_{md}, μ_{md}, and μ_{sd} which are attained by using the feedback gains K_{ma}, K_{md}, and K_{sd}, respectively, to alter the natural values.) The steady state response of F_s to an operator step input of magnitude F_H in each of these cases is

$$f_s = -\frac{K_{sp}}{K_{mp}} F_H \qquad \text{[BFR]} \qquad (7a)$$

$$f_s = -\frac{1}{K_{mf}} F_H \qquad \text{[IDF]} \qquad (7b)$$

Thus, if the slave contact force is to match the human input force in steady-state, $K_{mp}=K_{sp}$ in the BFR case and $K_{mf}=1$ in the IDF case.

3. STABILITY ANALYSIS

In formulating the results in this section, the reader should be aware that the system being examined is closed-loop only insofar as the operator input is unaffected by tactile feedback from the master arm and visual feedback from the slave arm. In reality, this is not the case, and we can expect the operator to add a significant amount of damping as well as inertia to the overall system. Our immediate conclusions on stability are therefore only valid for open-loop commands to the master arm when the slave arm is in contact with an object.

It is also important to realize that the actual transfer function for force contact is nonlinear, the simplest being a relay-type model. During experiments, the slave arm will typically bounce upon impact with an object, alternating between free-space motion and force contact. Thus the stability results are only valid for a "velcro" contact situation where the slave arm stays latched to the object. This situation is realistic for cases of fairly compliant surfaces but not for stiffer surfaces. Thus we cannot predict instabilities resulting from impact on a hard surface without resorting to a nonlinear analysis.

A root-locus stability analysis was done for the continuous-time system shown in Fig. 3. The root loci in the BFR case show the closed-loop poles varying as the gain on the master compensation, H_m, is increased from zero to infinity. Figure 5a shows the root loci for a bandwidth of 10 Hz in the BFR case when no natural damping is present ($\mu_m=\mu_s=0$) and the environmental stiffness is K_e=1000 N-m/rad where the "bandwidth" refers the system's natural frequency in the freespace case with gains determined by (4).

The conjugate pairs adjacent to the imaginary axis are largely the result of the interaction with the environment and become more oscillatory for larger K_e. These roots will be referred to as the "contact" poles. The roots along the circle close to the origin are the system poles for free-space motion, that is, they are the poles for τ_s/τ_h when no contact force is present, $K_e=0$. This circle increases in size with system bandwidth for constant damping ratios. These poles will be referred to as the "free-space" poles. The BFR case is always stable in the continuous-time case.

The root loci for the IDF case in Figure 5b are very similar to the BFR case except for the finite stability margin on the poles resulting from environmental contact. When K_{mf} is greater than unity in the zero damping case, the system becomes unstable. As expected, the faster response resulting from the use of direct force feedback over position/velocity error has the property of destabilizing the system for higher values of K_{mf}.

Adding damping has the effect of reducing the oscillation in the free-space poles but has little effect on the contact poles. Reducing the environmental stiffness leaves the contact poles still highly underdamped but lower in magnitude while the free-space poles remain relatively unaffected. For high enough reductions in stiffness, the free-space and contact poles may begin to interact. When processing delays are included, the stability is affected dramatically [7]. The system is very unstable for sampling frequencies less than 1000 Hz and environmental stiffnesses on the order of 1000 N-m/rad. Thus without taking into account the effect of the human operator, the system is found to destabilize rapidly when sampling and computational delays are present.

Without changing the model itself, attempts were made to incorporate the effect of the operator by adjusting two existing parameters: the master arm natural damping, μ_m, and the "environmental" damping, μ_e. A test was conducted in which the master arm was moved at a moderate speed until the slave arm impacted a wooden block. Experimental results are shown in Fig. 6 for an environmental stiffness of K_e = 1000 N-m/rad. As seen in the

velocity profiles, the master arm displays a minor backlash from the impact whereas the slave arm displays none at all. In the first simulation attempted, the master arm damping was increased to 10 kg-m^2/s and the impact induced instability; the initial bouncing at impact was less than in the undamped case, but then the oscillations began growing unbounded. When instead the environmental damping was increased to 10 kg-m^2/s, the position and velocity profiles shown in Fig. 7 exhibited fair agreement with the experimental results.

The contrasting behavior resulting from increasing the damping in the master arm versus the slave arm can be confirmed by replacing K_e in (5) and (6) by G_e

$$G_e = \mu_e s + K_e \tag{8}$$

and reconstructing the root-loci. Figures 8 and 9 give the results for increasing μ_m and μ_e, respectively, for the case shown in Fig. 5b. Increasing μ_m improves the gain margin by a factor of 20, but the poles still remain highly oscillatory and will destabilize under digital sampling. Increasing μ_e not only stabilizes the system for all values of K_{mf}, it also critically damps the contact poles.

Though the effects of the human operator are undoubtably more subtle than varying a value for the damping (the inertia is also modified, for instance), we can at least infer an interesting result from the previous analysis. It appears that the operator injects damping into the system not at the hand controller but at the slave arm. This is consistent with a fairly well known tenet in flexible body dynamics that collocated damping will always act to stabilize a system, but noncollocated damping may not. Thus it is with our system that damping at the point of contact will stabilize the bouncing caused by force impact.

4. EXPERIMENTAL RESULTS

The single-axis hand controller (SAHC) configuration shown in Figure 10 was used to test the control algorithms presented in the previous section. The apparatus consisted of a set of identical rotary hand controllers driven by a set of PUMA direct drive motors with a peak output torque of 5 N-m. Optical encoders with 3200-line resolution were used to measure the angular position of the motor shafts, and the electronics enhanced this resolution by a factor of four to give 16-bit accuracy in the angle. Strain gauges were mounted on each link and calibrated with weights to give torque data. Piezzo-resistive accelerometers (0-5g) were mounted at the base of the links near the handle to provide acceleration measurements in the IDF tests. Accelerometer and strain gauge signals were sent to a MicroVax via an A/D converter, whereas the encoder readings were sent to a counterboard prior to being read by the MicroVax off a DEC parallel line unit. After the control torques were computed, they were converted to analog drive voltages for the motor op-amps using a 12-bit D/A converter.

The SAHC was used in two different experimental configurations. The first series of tests run investigated the effects of master arm inertia and damping on the tracking ability of the operator. The second series tested the effect of using derived versus direct force feedback on the force perception of the operator. In the tracking experiments, a length of dowel was rotated about a fixed end aligned with the axis of rotation of the slave arm. The operator viewed the slave arm and tracking target through a television monitor (see Figure 11) and rotated the master arm to keep the slave arm following the motion of the dowel. The master arm dynamics were altered to enable four different inertias and two different values of damping to be felt by the operator. The target was also rotated at three different velocities for one set of inertia and damping values to investigate the effects, if any, of target speed on tracking ability.

Autocorrelation analyses of the master arm angular velocity were performed to determine the effects of changing these parameters. It was found that there were oscillations in the velocity, superimposed on a constant value (see Fig. 12). Variations in the amplitude and frequency of these oscillations were observed to depend on the modified inertia and damping of the master arm. The autocorrelation for the case shown in Fig. 12 is given in Fig. 13. In this example, the master arm had an apparent inertia of 0.0086 kg-m^2 and damping of 0.05 kg-m^2/s. Figure 14 gives the average frequency and amplitude of oscillations observed in the autocorrelations for different values of master arm inertia and damping. In all tests, the target speed was 90 deg/s and nonzero damping was 0.05 kg-m^2/s.

The effect of increasing the inertia of the master arm was to decrease both the frequency and amplitude of the oscillations. This is consistent with Gydikov [1] (refer to his Figures 4 and 5), although the actual values are somewhat different because of different hardware configurations. (In his experiments, the operator grasped a handle and rotated a flywheel which involved mainly wrist movement, whereas rotation of the SAHC involved movement

of the entire arm.) The effect of damping was not quite so apparent. For the same inertia, increasing the damping produced an increase in the average frequency of the oscillations although this was not statistically significant. Lower damping and target speeds did however produce significant increases in the average amplitude of oscillation.

The operator seemed to prefer higher master arm inertias when performing the tracking task, as they lessened the "flimsy" feel of the hand controller. Additional damping was not as important because the operator adds damping to the hand controller simply by grasping it. In fact, large values of damping made the motion feel sluggish and required more effort by the operator in controlling the rotations.

In the force experiments, the master arm was rotated until the end of the slave arm came into contact with a rotating circular cam (see Figure 15). The cam had a fixed offset, so that the task for the operator was to maintain a constant contact force on the oscillating surface. Gravity compensation in the control system ensured that the operator had to apply a continuous force to the master arm in order for the slave arm to remain in contact with the cam. The bilateral controller used force feedback derived from the angular position and velocity of the slave arm, and the impedance controller used a direct measurement of the force exerted by the slave arm on the surface. These two controllers were operated at bandwidths of 5 and 10 Hz and the cam rotation at 0.5 and 1 Hz.

In order to obtain a measure of the subject's ability to exert a steady contact force, the standard deviation about a nominal constant value (approximately 0.02 N-m) was calculated. This is illustrated in Figure 16, with the impedance controller (direct feedback) showing the least deviation in all tests and the 10 Hz bandwidth showing better results for both controllers. In all test runs, the average force felt by the operator before responding to the upward motion of the cam was 0.2 ± 0.02 N-m. The difference between the various scenarios was the time taken for this force to be felt, which led to a subsequent delay in the operator response. This delay involved both force perception and reaction time.

At a control bandwidth of 5 Hz, the operator showed little ability to maintain a constant contact force at any cam speed for either controller. For a cam speed of 0.5 Hz and bandwidth of 10 Hz, however, the subject was able to maintain a constant force over about 7/10ths of a cycle when using direct force feedback and 3/10ths of a cycle when using derived force feedback (see Figure 17). The operator's ability greatly diminished at the 1 Hz cam speed when using derived force feedback.

Although the data supports direct force feedback as a more accurate method of contact force perception, the operator could not distinguish between the two controllers during testing. This was because the stiffness of contact felt by the operator is set by the slave arm gains in the IDF case ($K_{mf}=1$) and by the cascaded viscous-spring combination of slave arm and master arm gains in the BFR case. Although the contact force may have felt the same in the two cases, the IDF feedback was exactly the contact force whereas the BFR feedback was corrupted by the secondary "spring" effect of the master arm.

The bandwidth difference, however, was very apparent. At 5 Hz the subject described the surface as feeling "spongy", while at 10 Hz contact was much more sharply defined. The harder contact felt in the 10 Hz case is attributable to the increased proportional/derivative gains used by the slave arm to follow the master arm. Since the PD gains for the slave arm affect the operator feedback in both BFR and IDF control, the operator will perceive a concurrent increase in stiffness of contact when the gains are increased.

5. CONCLUSIONS

The purpose of this research was to test the effect of a variable arm impedance and direct force feedback on the ergonomics and performance of a hand controller to be used in space teleoperation. Both arm inertia and viscosity were to be varied to test their effects on the operator's performance. Direct force feedback was tested against another well known technique from bilateral control which synthesizes a force reflection signal using relative position and velocity errors between the two arms.

Results from the tracking tests reveal a distinct operator preference for a frictionless, high-inertia hand controller. The desire for an undamped hand controller is attributable to the lower effort needed for locomotion. The preference for higher inertias is somewhat less clear, though part of the reason may be better impedance matching between the human arm and the hand controller. Hogan [8] shows that matched impedances provides the most efficient power transmission, a factor which may play the dominant role in an athlete's selection of a tennis racquet or baseball bat.

A less subjective reason may be found by observing the hand tremors present in the tracking tests. The operator found it more difficult to track the target using "lighter" hand controllers because it was more difficult to sense the motion. It is apparent that the subject used tactile information from the inertial forces as well as visual feedback to track the target. This tactile feedback may well be below the operator's force perception threshold to be of much use when using lower inertias.

The observation of inertia-dependent oscillations in the tracking tests yields more than ergonomic information. Their presence reiterates Gydikov's belief that the operator integrates errors and then exerts correcting forces using impulses. Bekey [9] takes this a step further in developing a finite-state model of the human operator. This approach warrants serious consideration, as it may lead to better approaches for modeling the human operator in closed-loop control tasks than the simple damping augmentation approach attempted in the last section.

The effect of direct force feedback was also a critical component of this study, and the force-contact tests yielded some interesting results. Improved performance with increasing bandwidth was not a particularly noteworthy result as it was primarily a function of the gains in the robot arm, not the feedback loop to the master arm. That the force perception was more accurate in the direct force feedback case was significant, however. Tracking a moving surface while attempting to maintain a constant contact force provided a way of quantifying this improvement. Interestingly, the operator was oblivious to his improved performance in the force feedback case. There is, however, a price to pay in stability for this improved performance. Since force feedback is analogous to an acceleration term in the control law, the feedback has a much higher effective gain than its position or velocity counterpart in bilateral control. This effect was easily verified by the root locus analysis, though the differences were somewhat less dramatic in the discrete-time case where the sampling period became a dominating concern [7].

This research showed several advantages of impedance control with direct force feedback over the popular bilateral configuration. These advantages are manifested in both operator perception as well as quantifiable performance measures. A hand controller with either automatic or operator adjustable impedances receiving direct force feedback information will be a strong candidate for space teleoperations involving dextrous manipulators.

ACKNOWLEDGEMENTS

This research was sponsored by NASA Goddard Space Flight Center under contract # NAS-5-28561 (Gary Mosier, project monitor). The authors wish to acknowledge Gary Mosier, Rick Schnurr, George Voellmer, Frank Bauer, and John Croft for their help during various phases of this study. Thanks also go to Reza Akhavan, Larry Alexander, and Tom Feiid for their assistance in writing the real-time code for running the hand controllers.

REFERENCES

1. Gydikov, A., "Sampling with Adjustable Frequency in the Hand Movement Control System," Transactions on Human Factors in Electronics, Vol. HFE-8, No. 2, June 1967 pp. 135-140
2. Paines, J. D., "Optimization of Manual Control Dynamics for Space Telemanipulation: Impedance Control of a Force Reflecting Hand Controller," Master of Science Thesis (SSL Rep. #20-87), Massachusetts Institute of Technology, Aug. 1987
3. Hannaford, B., and Anderson, R., "Experimental and Simulation Studies of Hard Contact in Force Reflecting Teleoperation," Proc. IEEE Int. Conf. on Robotics and Automation, San Francisco, April 7-10, 1986
4. An, C. H., and Hollerbach, J. M., "Dynamic Stability Issues in Force Control of Manipulators," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, N.C., 1987, pp. 890-896
5. Whitney, D. E., "Historical Perspective and State of the Art in Robot Force Control," Proc. IEEE Int. Conf. on Robotics and Automation, St. Louis, March 25-28, 1985, pp. 262-268
6. Eppinger, S. D., and Seering, W. P., "Understanding Bandwidth Limitations in Robot Force Control," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, N.C., 1987
7. Carignan, C., and Tarrant, J., "Impedance Control versus Bilateral Control: A Case Study in Manual Teleoperation," Technical Report, ST Systems Corp., Lanham, Md., Dec. 1988
8. Hogan, N., "Impedance Control of Industrial Robots," Robotics and Computer-Integrated Manufacturing, Vol. 1, No. 12, 1984, pp. 97-113
9. Bekey, G. A., and Angel, E. S., "Adaptive Finite-State Models of Manual Control Systems," IEEE Transactions on Man-Machine Systems, March 1968, pp. 15-20

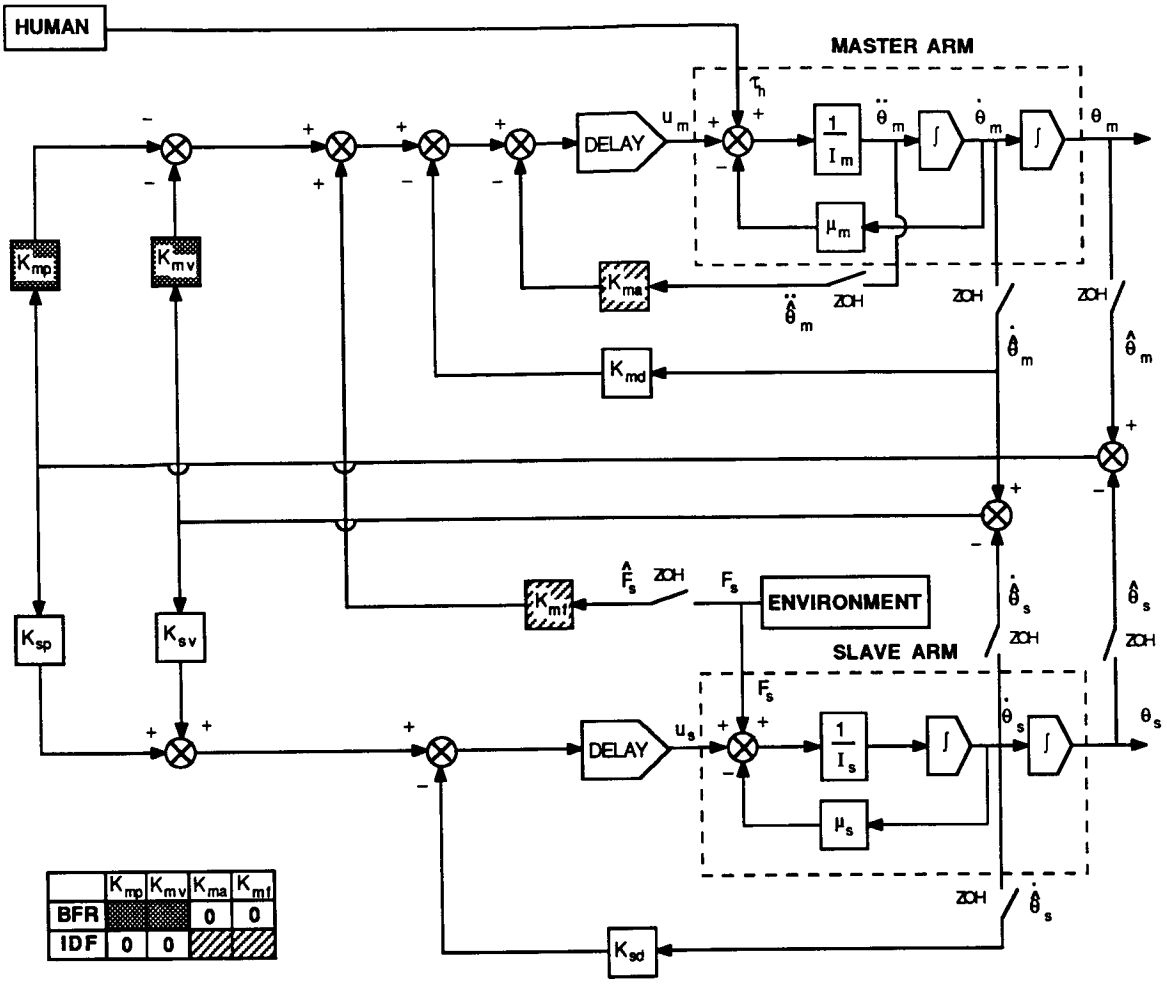


Figure 1: Force reflecting hand controller block diagram.

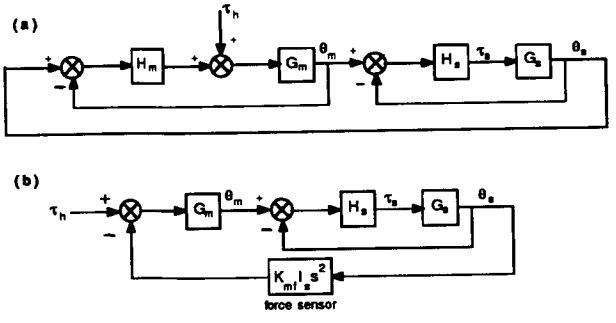


Figure 2: Block diagram for free-space motion in (a) BFR and (b) IDF cases.

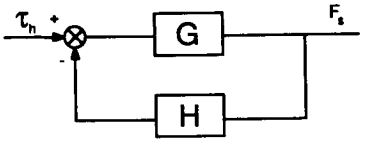


Figure 4: Reduced block diagram for force contact.

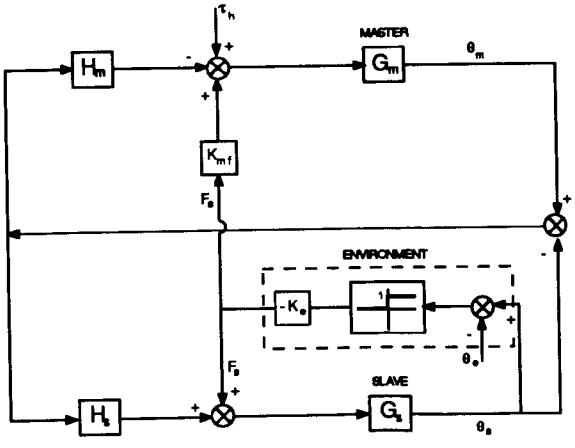


Figure 3: Block diagram for force contact.

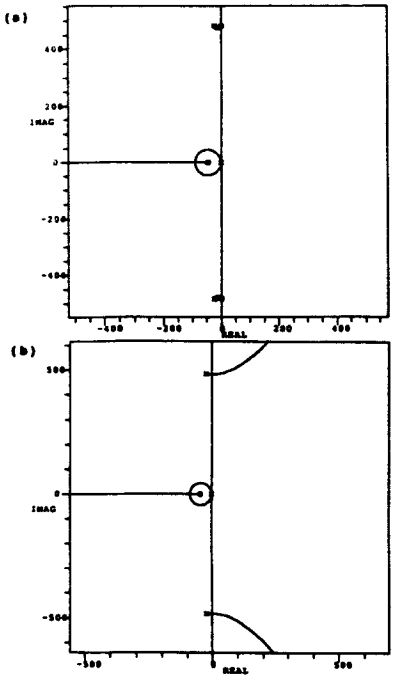


Figure 5: Root loci in (a) BFR case and (b) IDF case for 10 Hz bandwidth and K_e=1000N-m/rad.

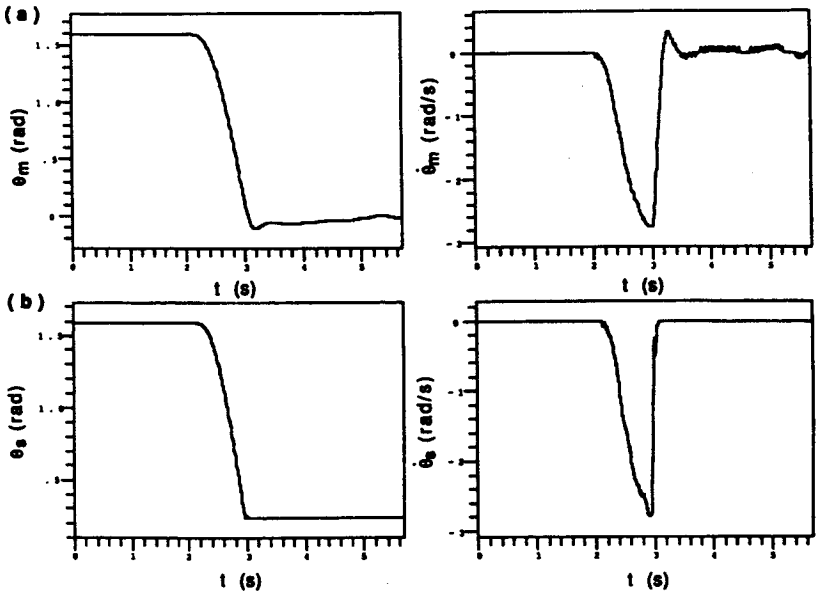


Figure 6: (a) Positions and (b) velocities of master and slave arms during impact experiment.

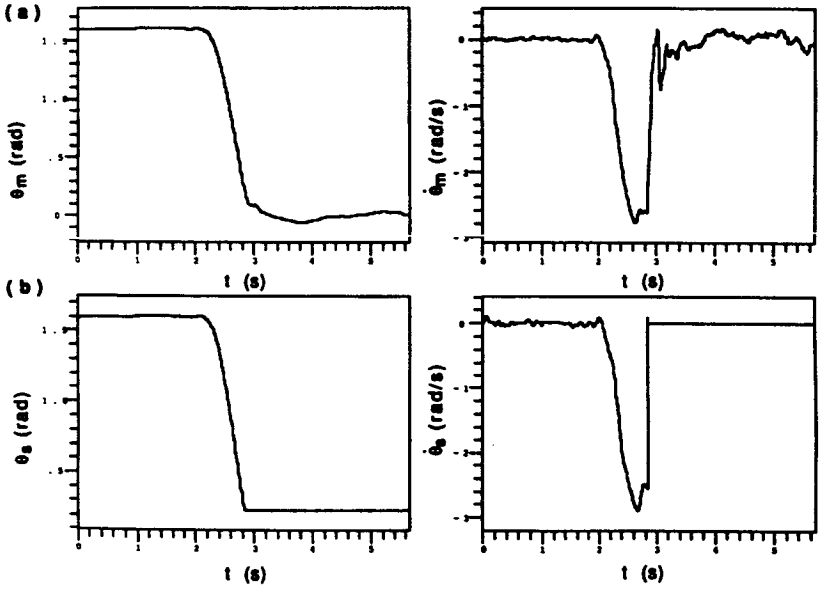


Figure 7: (a) Positions and (b) velocities of master and slave arms during damped environment impact simulation.

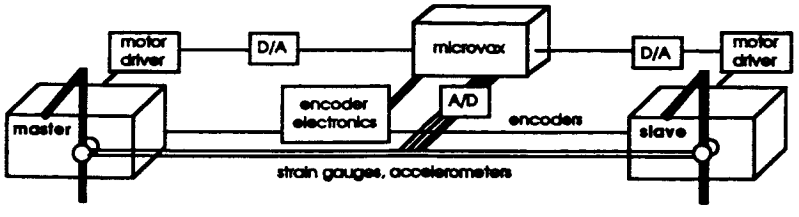


Figure 10: Hardware schematic of single-axis testbed.

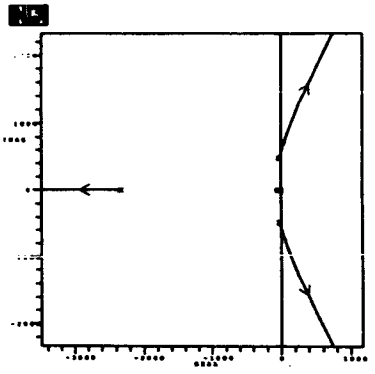


Figure 8: Root locus for master arm damping of 10 kg-m^2/s.

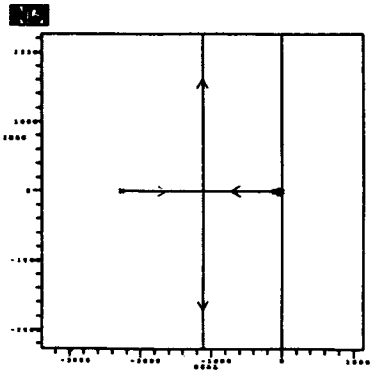


Figure 9: Root locus for damped environment (10 kg-m^2/s).

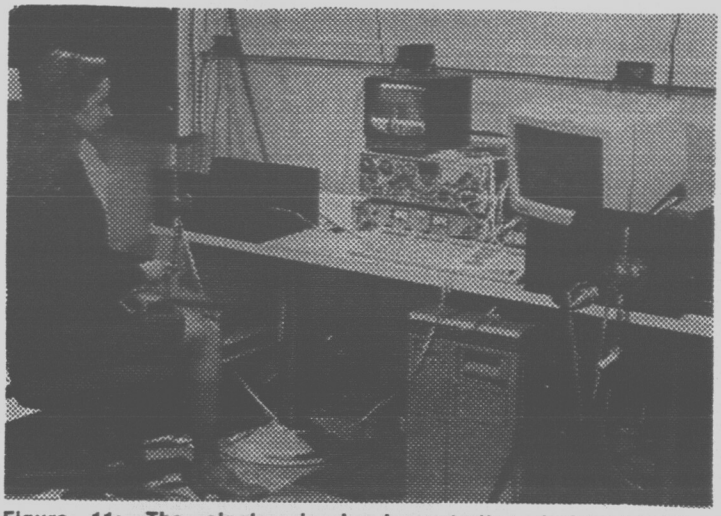


Figure 11: The single-axis hand controller during tracking experiments. The slave arm and tracking target (to right) were viewed by the operator through a television monitor (center).

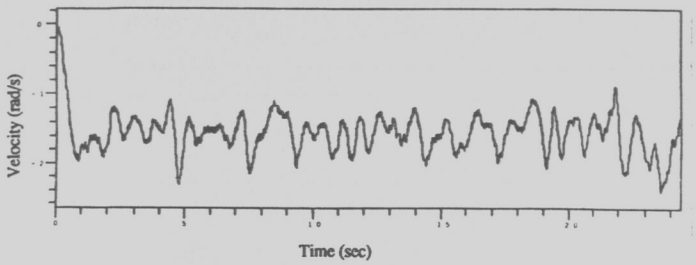


Figure 12: Velocity of master arm during tracking experiment (inertia 0.0086 kg-m^2,damping 0.05 kg-m^2/s).

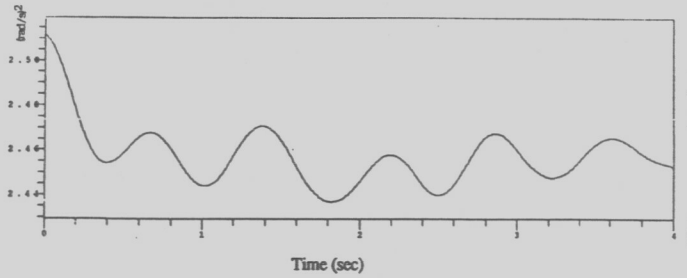


Figure 13: Velocity autocorrelation of master arm during tracking experiment.

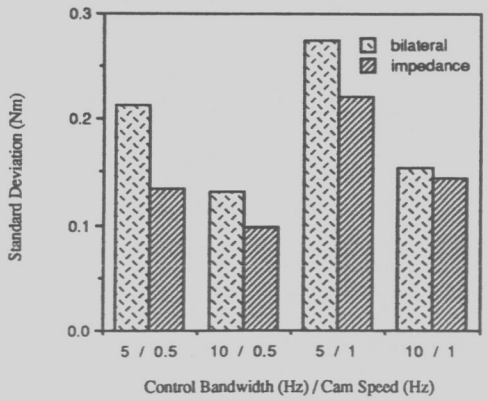
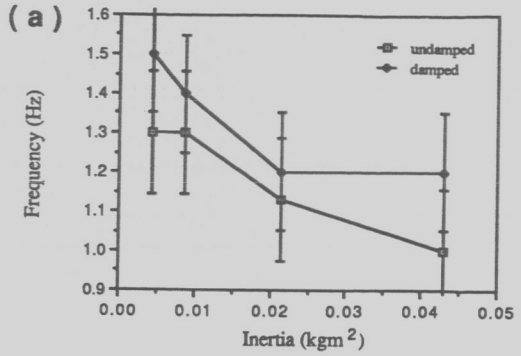


Figure 16: Standard deviation of torque exerted on the cam for various cam speeds and control bandwidths.



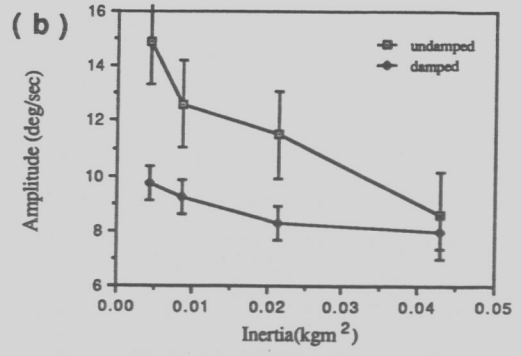


Figure 14: (a) Frequency and (b) amplitude of velocity oscillations observed in tracking experiments.

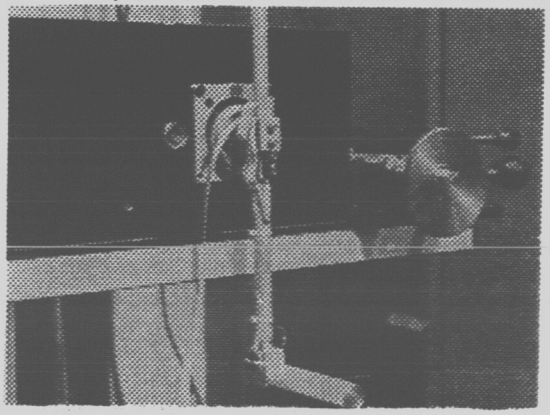


Figure 15: Slave arm and rotating cam (right) used in force experiments.

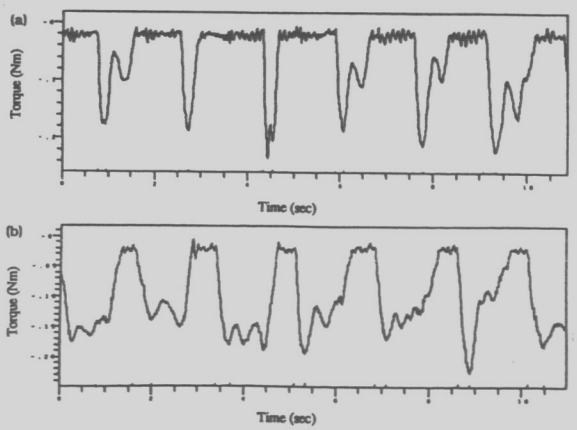


Figure 17: Contact torque on cam for (a) IDF and (b) BFR control for 0.5 Hz cam speed and 10 Hz bandwidth.

TELE-AUTONOMOUS SYSTEMS

TELE-AUTONOMOUS SYSTEMS: NEW METHODS FOR PROJECTING AND COORDINATING INTELLIGENT ACTION AT A DISTANCE

Lynn Conway,[1] Richard Volz[2] and Michael W. Walker[1]

[1]The Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.
[2]The Department of Computer Science, Texas A&M College Station, TX 77843.

Abstract - There is a growing need for humans to perform complex remote operations, and to extend the intelligence and experience of experts to distant applications. We assert that a blending of human intelligence, modern information technology, remote control and intelligent autonomous systems is required, and have coined the term "tele-autonomous technology," or "tele-automation" for short, for methods for producing intelligent action at a distance. Tele-automation goes beyond autonomous control in that it blends in human intelligence and action as appropriate. It goes beyond tele-operation in that it incorporates as much autonomy as is possible/reasonable. We discuss in detail a new approach for solving one of the fundamental problems facing tele-autonomous systems: the need to overcome time delays due to telemetry and signal propagation. We introduce new concepts, called time and position clutches, that allow the time and position frames, respectively, between the local user control and the remote device being controlled, to be desynchronized. The design and implementation of these mechanisms are described in detail. We demonstrate that these mechanisms lead to substantial telemanipulation performance improvements, including the novel result of improvements even in the absence of time delays. The new controls also yield a simple protocol for handoffs of control of manipulation tasks between local operators and remote systems.

1 Introduction

There is a growing need for humans to be able to perform complex, large scale, remore operations, and extend the intelligence and experience of experts to distant applications. This need is, perhaps, most dramatic in the area of space exploration. The National Commission on Space report *Pioneering the Space Frontier* [NAS86] describes in a vivid and exciting way the many potential scientific, commercial and colonization activities that could be accomplished in space over the next 50 years, and the Ride report *Leadership and America's Future in Space* [RID87] discusses specific missions that could be adopted to lead the nation into the new space era. These missions will require vastly more complex and larger scale remote operations than previous missions. There is an equally strong need for this capability, though not as dramatically obvious, for terrestrial applications. For example, undersea operations, mining, public safety, nuclear power maintenance, various defense applications and a wide variety of other hazardous operations would benefit strongly from an increased capability to perform complex remote operations. In this paper, we explore a new approach to achieving this type of operation.

1.1 Dimensions of the Problem Space

The terms "robotics", "tele-presence", "artificial intelligence" and "expert systems" appear throughout *Pioneering the Space Frontier* [NAS86] and the Ride report [RID87], and are indications of recognition of the need for complex, large scale remote operations. Yet, none of these technologies, either alone or in combination, are sufficient to satisfy the need for remote intelligent action. Robots,

artificial intelligence and expert systems are essential for building autonomous systems, while tele-operation and tele-presense are directed toward making the sensory-motor coordination of a remote robot lie entirely within the brain of a human operator. Pure tele-operation, however, is too awkward and human intensive to be fully effective. Communication time delays to/from remote operations further complicate the control. On the other hand, while AI technology is capable of fairly intelligent cognitive activities, it lacks the capability to manage most real-time manipulation tasks. Such approaches also do not adequately utilize the human intelligence that may be present at the remote site.

Many opportunities have been overlooked by concentrating on total automation, artificial intelligence, or simple tele-manipulation alone. New, more generalized thinking is required.

We assert that one must return to the fundamental problem space, and examine all possible cross products of its dimensions to determine new, more effective solutions. The sketch of Figure 1 shows our view of this basic problem space. An element of intelligent activity is an entity at a specific location performing a process (perception, cognition, or action), represented as a point in this space. Intelligent activity is a symphony of these elements connected by processes of communication, much the same way that language is formed of the elements of grammar and the rules governing their use. The space is discrete, and a single entity may be represented by more than one point (but all on a single location plane) if it performs more than one process. These dimensions imply a wide range of systems that are possible. Many kinds of devices, not just robots and vehicles, are possible. Manned operations are included. Perception, cognition and action may be divided or shared among entities. Cooperating entities may be at different locations. Cooperative relations among the entities may be dynamic. For example, tele-control of a device may be handed off between different humans in different locations. And, groups of humans and machines at different locations may be able to dynamically cooperate. For example, in one future scenario, humans at scattered locations – e.g., on Earth and in the Space Station – may cooperate on a repair satellite task using remote robots, as sketched in Figure 2.

These dimensions also help place the traditional approaches in perspective. For example, a standard tele-manipulation system would have two points in the human plane indicating perception and cognition, and a single point in the manipulator plane indicating remote action. An autonomous system would be a set of three points in the plane of some device. The traditional approaches are thus seen as constrained to a set of three points lying in one or two planes perpendicular to the "form" axis. Tele-robotics relaxes the constraints by allowing both the human and robot plane to have points of the same process type, indicating a sharing of the process, most commonly cognition. However, these improvements from tele-robotics are still quite limited in comparison to the full space of possibilities.

We suggest that to achieve the most useful control configurations, a blending of modern information technology, human intelligence, remote control, and intelligent autonomous systems is required. When we first began work in this area, we coined the term "tele-autonomous technology" [CON87a, CON87b], or "tele-automation" for short, for the new methods for producing intelligent action at a distance, in order to emphasize the interactions of humans with remote, intelligent, partly-autonomous systems. We also envision tele-autonomous activities as typically involving several humans and several partly autonomous systems in coordinated activities. Thus the new tele-autonomous system technology will also blend in the methods of "collaboration technology" or "computer supported cooperative work" for effective goal-seeking coordination in such multi-agent systems [GRE88].

Tele-automation represents a new way of viewing and developing systems that must perform intelligent action at a distance. The remote systems will be as intelligent and autonomous as possible/appropriate, but capable of being guided when necessary by humans. This can greatly reduce the need for continual human involvement, while complementing the powers of AI autonomous systems. Machine learning can provide the capability for the autonomous part of the system to gradually take

over more and more of the operation time the human generates inputs to the system. Collaboration technology can provide the ability for cooperative interactions.

Of course, the intelligent agents at the remote location need not be autonomous devices, but might well be intelligent, but non-expert, humans. Many operations, such as infrequent maintenance operations or actions ensuing from unforeseen events, will be difficult or impossible to automate. Tele-autonomous technology will allow experts in one location to guide non-experts in another in much more effective ways than currently possible. The payoff in terms of reduced training costs (many astronauts trained for months to perform a 45 minute repair on Solar Max [ESS85]) and the extension of operations that can be performed in space will be tremendous.

1.2 A New General Infrastructure for Remote Operations

The future NASA missions described in the Commission on Space and the Ride reports depend heavily upon performing extensive, labor intensive intelligent actions in space. Examples are exploration, remote sensing, surveying, prospecting, mining, manufacturing, assembly, payload maintenance and servicing, agriculture, experimentation, and many routine daily operations. Details of new requirements for such remote systems are found, for example, in NASA's Automation and Robotics Progress Reports [NAS88]. It would be extremely costly if humans were to directly perform all these operations. Even if these costs could be borne, there are real constraints on the number of humans that it will be possible to sustain in space for the next several decades, thus limiting both the amount of labor and expertise that can be directly resident.

In the past, the research community has proposed "solutions through automation". However, in recent years it has become clear that automation offers only a partial solution because of its inability to function robustly when dealing with spontaneous, unexpected events. While artificial intelligence promises to ease some of these "automation difficulties," we note that the advances of AI have been primarily in the area of machine cognition. AI does not yet offer comparable advances in the area of machine perception and action. While current and near future technology offer us "intelligent thinking machines," they offer us only very limited means for providing "clever perceiving and manipulating machines."

The whole future of space exploration thus depends upon the creation of some form of infrastructure for bringing a *mixture* of human and machine perception, thinking and manipulation skills to bear on the many tasks to be done, even though those skills may be scattered over large distances in space and time. While less obvious because the cost of human labor is not so high, the same infrastructure can have enormous payoff for many of the same applications in a terrestrial setting as well, particularly those applications involving hazardous or difficult to reach environments. Rather than foreseeing complete automation solutions to these problems, we instead visualize construction of an infrastructure that enables humans to collaboratively project and focus their capabilities to perform distant tasks.

Of course, it will be important to provide the means to evolve toward more fully autonomous systems where that is feasible. We visualize the provision of new forms of machine learning technology that can mimic and learn oft repeated skills, thus relieving humans from having to perform tasks that have become somewhat routine.

But the key point is that a new form of "work infrastructure" is essential to the rapid and efficient exploration and development of space, and will be highly useful to many terrestrial applications as well. We believe that tele-autonomous system technology is the basis for that new infrastructure.

2 Overview of Tele-autonomous Operation

Tele-automation represents a new way of viewing and developing systems that must perform intel-

ligent actions at a distance. It goes beyond autonomous control in that it blends in human intelligence and action as appropriate. Tele-automation goes beyond tele-operation in that it incorporates as much autonomy as is possible/reasonable. Tele-automation is also concerned with enabling collaboration among multiple human and autonomous systems, and with enabling adaptation, or machine learning, by and among the autonomous systems.

Figure 3 shows, at a conceptual level, the structure of a single local/remote pair in a basic tele-autonomous system. The spatial reference frame is taken to be that of the human controller at the left side of the figure, i.e, the controlled environment is remote. The controlled environment can include humans and/or any manner of device. The remote intelligent controller receives data from multiple sensors, and provides multiple outputs, encompassing anything from servo level control signals to a robot joint, to video signals to a heads-up display worn by a remote human.

The inputs on the local side of the system may be any form of input control by the human, from simple joystick control, to complex cockpits with many inputs, to discrete commands for the remote controller to perform complex tasks. The local display represents any kind of feedback to the human about the remote environment. This will include both simulated information and actual feedback signals and may be composed of TV images, complex graphics, force reflection on input devices, or even high speed data analysis. The distance between the local and remote sites can produce substantial time delays in the signal transmission between them.

Tele-autonomous control of even a single local/remote controller pair provides many operating modes, including:

1. Direct continuous tele-operator control of a remote device. The remote controller merely follows its inputs. This is currently the most common form of operation.

2. Shared continuous tele-operator control of a remote device. The remote controller performs higher than position serving. For example, it might treat received inputs as being relative to an object to be manipulated and perform appropriate transformations before following them [VOL88]. Or, it might treat received inputs as a nominal path, and perform some local sensing and replanning to reach the goals of the nominal plan.

3. Discrete command control by the human operator of the remote device. This implies a higher level of capability in the remote portion of the controller that can vary from simple setpoint control of a number of satellite antenna positioning servos, to complex task analysis, planning and execution. At this level the commands become highly task specific, though the lower level primitives utilized may be more generic.

4. Supervisory control[3]. The remote device operates in a largely autonomous mode and only interacts with the human when it encounters a situation it cannot handle, i.e., management by exception, or in which the human notices an opportunity to improve performance, i.e, opportunistic management. It differs from the discrete command mode principally in the frequency of interaction with the human controller, and the philosophy of being largely autonomous. One local human operator might supervise a fleet of remote devices.

5. Learning control. The remote controller is given an intelligence that allows it to learn from human inputs and sensor information, and subsequently deduce correct behavior in similar situations without human intervention.

6. Guidance of remote non-expert humans by local experts. In this mode a variety of media, visual displays, graphics, touching, pointing, etc., are used to achieve a collaboration between the local expert and the remote non-expert.

[3]We use the term supervisory control to describe a much higher level mode than that usually attributed to the term. However, our usage fits the intuitive interpretation if the term quite well.

Groups of such basic systems, possibly with local controllers in different locations, will make up larger scale tele-autonomous systems. Many kinds of interactions will be possible, from hand-offs of control between different local control agents (even if in different physical locations) to shared cooperative action of the remote devices.

3 Basic Tele-automation Controls

Fully general tele-autonomous systems do not yet exist, and will be the subject of research for a long time. However, we have recently discovered some fundamental principles that we believe will be part of the architectural foundation of almost any general tele-autonomous system.

One of the most fundamental problems facing tele-autonomous systems is time delay due to telemetry and/or signal propagation delays. Even modest time delays have long been known to cause instabilities in control systems such as robots. And, the time delays present in space applications are anything but modest.

We review here a sequence of interface control concepts originally presented in [CON87a] that collectively underlie efficient control of manipulation tasks and also enable simple protocols for exchange of such tasks among control agents.

3.1 Coping with Time Delay

Although tele-manipulation has been studied for years (e.g., see [GOE52], [KUG72], [HIL79], [DRA87], [MOL87]), Noyes and Sheridan [NOY84] were the first to make significant progress on the tele-manipulation time-delay problem. Noyes and Sheridan suggested that the operator control a local simulation of the telerobot, with the control signals then sent in parallel to the simulation and the remote telerobot. The simulation is then displayed superimposed over the return video. In this way the operator can "see" the effects of the control immediately without having to fully wait for the return signal from the telerobot. This system concept is sketched in Figure 4. In the system we built to test the concept, we used a model of the telerobot on the IRIS workstation, making it easy to simulate time delays and easing solution of the correspondence problem between the simulated and actual robots.

Figure 5 presents a visualization of telerobotic manipulation using a forward simulation to cope with the time delay. The wire frame is the forward simulation that directly responds to operator control, and the solid frame represents the time delayed image of the real telerobot. Much faster and smoother control is achieved. Task time may be reduced to nearly that of the no-delay case, as shown in Figure 6. This is a first step towards evolving machine manipulation visualization, since the visualization could help cope not only with communication delays, but also with computational delays within a self-contained autonomous agent.

3.2 The Time Clutch

In the work of Noyes and Sheridan described above, the time frames of the simulation and the robot are separated by the time delay of the telemetry and propagation. However, there is no intrinsic reason to maintain this synchrony. We thus introduce the concept of a "time clutch" that can disengage synchrony between operator specification time and telerobot manipulation time during path specification. Our hypothesis is that operators can generate a path faster than the robot can follow it. This is particularly true of large space telerobots such as the Remote Manipulator System (RMS) [NAS81]. Once generated, a path segment can then be followed more quickly by the robot than would be the case if the robot were time-synchronized to the specification process; with time synchrony disengaged, the robot can steadily proceed at nearly its maximum rate, subject of course to error limits and hard constraints.

Figure 7 shows a path being generated well out in advance of the actual robot by an operator using forward simulation with time clutch disengaged. The performance of an operator when using the time clutch while performing the task of touching a series of boxes in our experimental trials [CON87b] is shown in Figure 8. Remarkably, the performance is better than control without the time clutch even in the case of no time delay.

This step in the evolution of machine manipulation visualization enables the cognitive agent to "look and think ahead" of the manipulation under control, with the look-ahead time being elastic, and not just a fixed internal or external system time delay. The implementation of this new capability requires only a simple mutation of the forward simulation previously used for coping with a time delay.

3.3 The Position Clutch

We next introduce the concept of a "position clutch" which enables a disengagement of position synchrony between simulator and manipulator path. We hypothesize that faster, shorter, cleaner paths can be generated on difficult tasks using this control. This idea is illustrated in Figure 9, which shows the use of the position clutch to disengage from path generation during a close approach to a difficult manipulation (in this case, touching a small object).

Suppose, for example, that the operator had arrived (in the simulation) at point A ahead of time by using the time clutch. The position clutch can then be disengaged, stopping the output from the operator control from going to the real telerobot – it will only go to the simulation. When the forward simulator is in good position, the position clutch will be reengaged, causing a short, smooth path to be inserted that links to the earlier path. This avoids inclusion of jittery prepositioning movements in the final path to be followed. Further, the time spent by the operator in achieving the proper position will not be incurred by the real telerobot since these motions were "clipped" out of the path sent to the telerobot.

The operator has thus used up some of the time saved through use of the time clutch, with the result that the overall task time of the telerobot is reduced still further. This level of manipulation visualization corresponds to quick visualizations and visualized trials of multiple alternatives prior to commitment to action, and its implementation requires only another simple mutation of the basic forward simulation capability.

3.4 The Time Brake

To handle contingencies and errors we introduce the concept of a time brake. This control can be used to deal with situations such as something falling over a previously generated path, as illustrated by the "X" in Figure 10. In Figure 10 we see the time brake being applied and the forward-simulated manipulator backing down the path (in a race to get on the other side of the obstacle before the real system gets there).

This aspect of visualization corresponds to seeing something about to happen that will interrupt an action previously visualized but not yet underway. If it had gotten underway, or is allowed to get underway, the system will have to deal with it through local reflex action or crash. But, if visualized in time, the cognitive agent can withdraw the action using the time brake.

3.5 Task Handoffs and Rendezvous

These basic tele-autonomous system interface controls also provide the basis for a simple, elegant protocol for hand-offs and rendezvous of tasks between different control agents. Imagine two operators, one in control of the telerobot and the other about to take over in relief of the first, as sketched in Figure 11. Each operator would be in control of a simulation of the telerobot, but only the control

signals of the first would be sent to the real telerobot. The relief operator would, with position clutch disengaged, guide his/her simulation as close to the first operator's as possible (or required). The first operator then disengages their position clutch, leaving the path "hanging". Figure 12 shows this moment in the interaction.

The second operator then engages their position clutch, rendezvousing with the path and taking control of future path generation. When the actual manipulator passes over this path segment, it will do so smoothly and will not notice that a change of control agent has occurred in mid-maneuver. We can again find interesting biological analogies to this visualization situation. For example, consider the interactions among basketball players as they previsualize fast-paced multiplayer interactions.

We believe that this simple protocol can be built upon to mechanize quite a wide range of manipulation interactions between autonomous agents.

4 Future Directions

Tele-autonomous technology presents new challenges in human computer interaction. We have proposed a set of interface controls that are conceptually simple and easy to mechanize. The controls are generic ones that may be applicable in many different specialized situations. They are also cognitively and manipulatively accessible to the uninitiated by analogy. But many other new human interface aspects haven't been pinned down at all. How is the operator to visualize where they are, who has control of what, and who they give control to next as they enter or leave some subtask within a complex task lattice? What measures can we provide concerning operator performance, and what feedback can we provide? And what about the analysis and design of cognitive and manipulation tasks themselves? Research can perhaps provide better measures of joint human-machine cognitive-manipulative performance. Analyses similar to those in [CAR83] may then lead us to design intermixings of human and machine activity that yield substantial improvements in overall performance.

The work poses some additional new challenges in robotics, such as the eventual need to perceive, model and forward simulate not only the remote tele-automaton, but also portions of the remote environment itself. Forward simulation will work fine when interacting with static objects, but what about interactions with moving objects? The simulation based methods we have discussed are dependent, in pure form, entirely upon the quality of the robot and environment models available and the accuracy with which tasks must be performed. In all of our experimental tests to date, the accuracy required was well below the accuracy of the models, and this was not a problem. However, most assembly tasks involve contact among the parts and have much higher accuracy requirements. Moreover, independent of accuracy requirements, even small errors when contacts are involved can produce very high, possibly damaging, forces. Solutions for this important class of problems is essential for many, if not most, applications. [VOL88] describes these basic problems in greater detail and outlines a number of possible directions for solution.

Further work is needed on methods for path-error specification and associated methods for the time optimization of path following, such as in [SUH87]. Additional work is also needed on autonomous "reflex" actions that the remote robot can perform when encountering uncertainties (particularly those involving contact) not modeled in the forward simulation. We also see the need for augmented AI programming environments that interface in such a way with real-time programming environments as to easily enable rapid estimation of time available for short-term AI planning tasks (enabling us to select among AI methods as a function of available time).

Exploration of different dimensions of tele-autonomy is also likely to lead to near term advances of considerable utility. In particular, most of the base technologies exist for developing what we have called "remote coaching" systems in which a local expert can coach a remote technician in complex experimental or maintenance tasks. A prototype remote coaching system is described in [WAL88].

This prototype includes a modest, but extensible, expert system in the remote controller that can help the technician with most problems and call in the expert when needed. The system allows graphic or image data objects to be selected from a library and placed on workstation screens for both the technician and the expert. The expert has a graphic capability for drawing on both screens, as well as being in voice contact with the technician. Even slow scan video is available. Work is still needed on the most effective means of on-line interaction, however, such as simulation video and graphics, and the extension of collaboration technology to the domain of cooperative manipulation tasks.

5 Conclusions

We have pointed out that there is a growing need in many areas of our society to be able to achieve remote intelligent action at a distance, and that traditional methods of automation and artificial intelligence are inadequate for such tasks. We have further introduced three dimensions that characterize the problem space: (i) the type of process being performed (perception, cognition, and/or action), (ii) the form that is performing the process (human, robot, automatic vehicle, etc.), and (iii) the location at which the process is being performed. We have coined the term tele-autonomous systems to describe systems addressing this problem space. Tele-autonomous systems are represented by a set of points in this space spread across more than one location plane.

One of the most fundamental problems that must be overcome in building such tele-autonomous systems is time delay resulting from telemetry or signal propagation. Simulation of remote devices and environments is part of the solution. We have introduced the notions of time and position de-synchronization (implemented through time and position clutches) to allow the simulation to be operated faster than real time and to permit an on-line "motion editing" to be achieved. Our early experiments involving the use of such time and position clutches suggest that dramatic improvements in performance can be achieved through the use of these clutches, even when there is no time delay. Moreover, the time and position clutches can be used to accomplish a new interaction protocol for hand-offs between two agents controlling a remote device. This protocol is based upon a shared visualization of the intended motion of the device.

While the number of potential applications for tele-autonomous systems is immense, there is yet a great deal of research to be done. We concluded by identifying just a few of the areas needing research. Among the more important were the extension of the time and position clutch ideas to situations involving precision contact among the objects involved, the development of "remote coaching" systems, learning systems, and the development of collaboration technology to support group manipulation tasks.

We believe that tele-autonomous systems research can yield methods and systems for improved projection of intelligent action at a distance in time and space. This interdiscipline presents interesting new research opportunities to teams having expertise in robotics and automation, artificial intelligence, and the psychology of human-computer interaction. We envision many possible applications for the resulting technology, not only in space and defense systems, but also in design systems, production systems, and eventually in personal and recreational environments.

6 Acknowledgements

We gratefully acknowledge support from the Research Excellence Fund of the State of Michigan, support provided by the grant of a high performance IRIS graphics workstation from Silicon Graphics, Inc., and the contributions of Lee Hagerhorst and Lejun Shao to the rapid prototyping of our first experimental system.

7 References

[CAR83] S.K. Card, T.P. Morand and A. Newell, *The Psychology of Human-computer Interaction*, Lawrence Elbaum Assoc., Hillsdale, NY, 1983.

[CON87a] L. Conway, R. Volz and M. Walker, "New Concepts in Tele-Autonomous Systems", *Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program*, March 11, 1987.

[CON87b] L. Conway, R. Volz and M. Walker, "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotics Technology", *Proceedings of the IEEE International Conference on Robotics and Automation*, March 30, 1987.

[DRA87] J.V. Draper, J.N. Herndon and W.E. Moore, "The implications of force reflection for teleoperation in space", *1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, May 1987.

[ESS85] Essex Corporation, "The Solar Max Repair Mission", Final Report to NASA, Contract NAS5-27345, Essex Report No. N-85-05, June 25, 1985.

[GOE52] R.C. Goertz, "Fundamentals of general purpose remote manipulators", *Nucleonics*, 10:36-42, November 1952.

[GRE88] I. Greif, Ed., *Computer-Supported Cooperative Work: A Book of Readings*, Lotus Development Corp., Morgan Kaufmann, New York, May 1988.

[HIL79] J.W. Hill, *Study of modeling and evaluation of remote manipulation tasks with force feedback*, NASA Technical Report CR-158721, July 1979.

[KUG72] D.A. Kugarth, *Experiments evaluating compliance and force feedback effect on manipulator performance*, NASA Technical Report CR-128605, August 1972.

[MOL87] J. Molino, "Robotics development facility preliminary engineering results", *Flight telerobotic servicer in-house phase-B study, First NASA/Industry Briefing*, December 1987.

[NAS81] NASA, "Shuttle Flight Operations Manual, Payload Deployment and Retrieval Systems", Vol. 16; Flight Operations Directorate, Johnson Space enter, June 1, 1981.

[NASA86] *Pioneering the Space Frontier*, U.S. National Commission on Space, NASA, Library of Congress, 1-211, May 1986.

[NAS88] *Advancing Automation and Robotics Technology for the Space Station and for the U.S. Economy: Progress Report 6 - October 1987 Through March 1988*, NASA Technical Memorandum 100989, June 15, 1988.

[NOY84] M. Noyes and T.B. Sheridan, "A Novel Predictor for Telemanipulation through a Time Delay", *Proc. of the Annual Conference on Manual Control*, NASA Ames Research Center, Moffett Field, CA, 1984.

[RID87] S.K. Ride, *Leadership and America's Future in Space*, NASA, August 1987.

[RTI82] RTI, *RTI Force Sensing Wrist User's Manual*, Robot Technology, Inc., Los Altos, CA, 1982.

[SUH87] S.H. Suh and A.B. Bishop, "Tube Concept and Its Application to the Obstacle Avoidance Minimum-time Trajectory Planning Problem", Univ. of Michigan Robotics Laboratory paper submitted to the *IEEE Journal of Robotics and Automation.*

[VER86] J. Vertut and P. Coiffet, "Teleoperations and Robotics: Applications and Technology", *Robot Technology*, Vol. 3B, English Trans., Prentice- Hall, 1986.

[VOL88] R. Volz, L. Shao, M. Walker and L. Conway, "Tele-Autonomous Control Involving Contacts", Technical Report, RSD-8-88, University of Michigan, Robotics Research Laboratory, 1988.

[WAL88] M. Walker, S.-Y. Sheu, R. Volz and L. Conway, "A Low Cost Portable Tele-Autonomous Maintenance Station", presented at the USAF/NASA Space Operations Automation and Robotics Workshop ("SOAR '88"), Dayton, OH, July 1988.

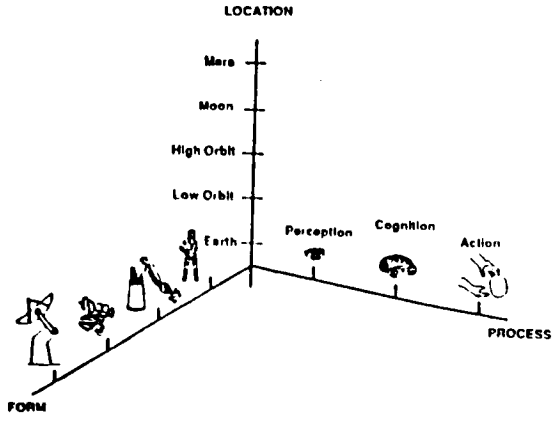


Figure 1: Dimensions of the problem space of intelligent action at a distance.

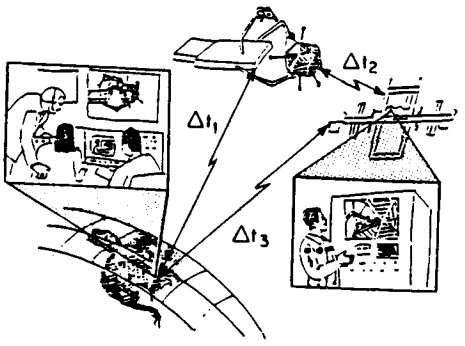


Figure 2: Cooperative repair of a satellite by mission specialist in the Space Station and experts on Earth.

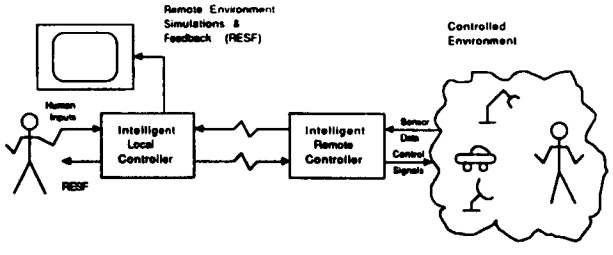


Figure 3: Basic components of a tele-autonomous system.

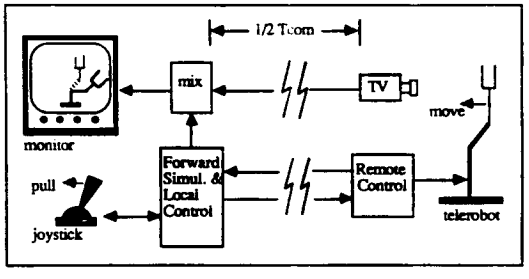


Figure 4: Using forward simulation and predictor display to cope with time delay (after Noyes and Shridan [NOY84]).

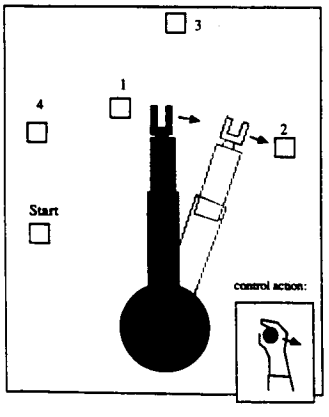


Figure 5: Visualizing manipulation through a time delay using forward simulation.

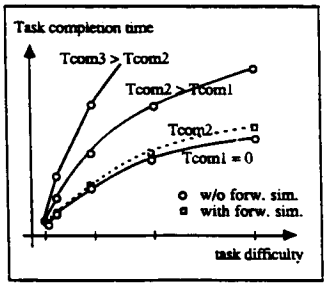


Figure 6: Task completion time as a function of task difficulty and communication delay, showing performance improvement using forward simulation.

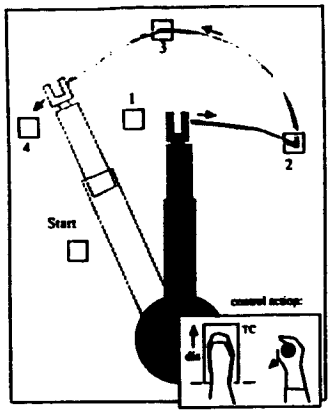


Figure 7: Rapid manipulation path generation using forward simulation with time clutch.

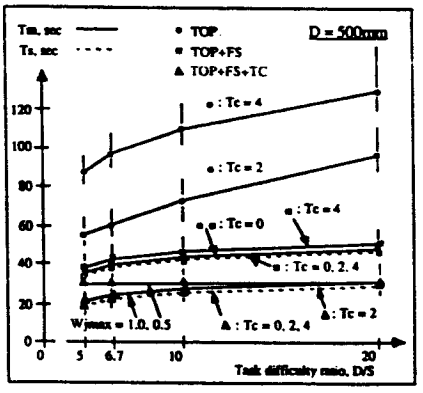


Figure 8: Initial trail results, showing Ts,Tm as functions of system and task parameters for three models of control.

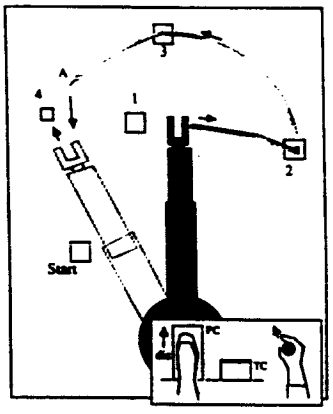


Figure 9: Using the position clutch to cope with a more difficult manipulation.

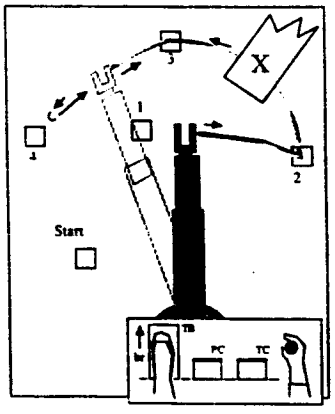


Figure 10: Using time brake to handle a contingency.

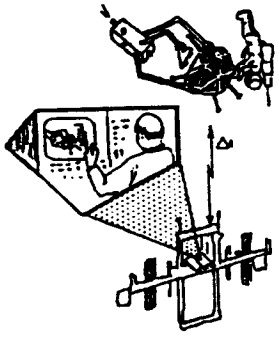


Figure 11: Application of manipulation task handoff through a time delay, using tele-autonomous system controls.

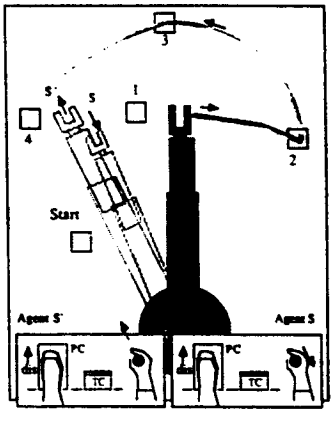


Figure 12: Using time and position clutches to handoff task to another forward simulation agent.

AN ADVANCED TELEROBOTIC SYSTEM FOR SHUTTLE PAYLOAD CHANGEOUT ROOM PROCESSING APPLICATIONS

M. Sklar, Ph.D., D. Wegerif, Ph.D.
MCDONNELL DOUGLAS SPACE SYSTEMS CO. - KSC DIVISION

L. Davis
NASA - KENNEDY SPACE CENTER

ABSTRACT

Ground processing of the Space Shuttle and its associated payloads at Kennedy Space Center is extremely time consuming and costly. Automation of both physical and information system processing can significantly reduce costs, processing time, and operational hazards to human technicians and space hardware, and improve reliability. Extensive efforts by NASA-KSC and its associated shuttle and payload contractors to implement automated systems are now ongoing. One particularly attractive application, which is a crucial operation in need of improvements, is payload and shuttle processing at the Payload Changeout Room (PCR). The PCR located at each of the two shuttle launch pads is a large clean room mounted on the Rotating Service Structure. All payloads are partially processed, accessed and in some cases transferred to the shuttle bay from this room.

Unfortunately, the current handling mechanism and platform system does not provide completely flexible access to all payloads and critical Shuttle bay locations. Thus, either human technicians are placed in hazardous positions, or specialized fixtures, scaffolds and lifting devices are used. These alternatives all increase cost, possibly increase payload exposure to potential damage and reduce efficiency, flexibility and overall cleanliness. Thus, to potentially alleviate these inherent difficulties, a teleoperated, semi-autonomous robotic processing system for the PCR is now in the conceptual stages.

A clean room manipulator arm, custom designed for the PCR, will perform basic processing tasks such as inspection, insertion and removal of small items. A highly redundant avoidance system will be incorporated to guarantee that collisions with delicate space hardware are avoided. A redundant arm (greater than 6 DOF) will likely be required due to the large workspace which is extremely cluttered and constrained when a payload is in the shuttle bay or PCR. The system will be driven by high-level user-entered commands or through a manually operated joystick. Thus, a complex planning and reasoning system based on Artificial Intelligence technology will be required. The primary challenge in this project is the integration of leading edge automation technologies now available. The integration and further development of the required technologies is now being accomplished through a joint NASA JPL/KSC demonstration program which began in 1988. This 3 year program will demonstrate actual PCR tasks using a full size mock-up, actual flight hardware and the ASEA IRB90 robot arm located at KSC's Robotic Applications Development Laboratory. The complete PCR robotic system as currently conceived is described here. Critical design issues and the required technologies are discussed.

1. INTRODUCTION

A number of ground processing operations at KSC could be improved through automation. Recently a study was conducted to determine where automation may benefit ground processing of payloads including the Space Station, [1]. The reasons for introducing automation include the reduction of hazards to payloads and processing personnel, reductions in operational costs and processing time. In addition, automation will help to improve processing reliability, verification and documentation.

One of the best examples of physical processing automation at KSC is the cleaning and refurbishment of the Solid Rocket Booster's nose cone, instrument module, and aft skirt. For this application, the justifications are easily identified: the

process is repeated for two SRBs for each mission, it eliminates the exposure of humans to a hazardous environment, produces more consistent results, and has provided a significant cost savings to SRB refurbishment [2].

For most of the automation applications needed for payload processing, full autonomy is not possible due to current state-of-the-art capabilities of robots and controllers. Instead of totally replacing the human, his physical skills and capabilities should be augmented through automation. Most payload processing robotic applications, require a balanced combination of computer intelligence and human reasoning and control. The Payload Changeout Room (PCR) is an ideal candidate for robotic technology due to the time constraints, criticality of operations, current difficulties associated with payload access, cleanliness requirements and associated cost of operations.

The major challenge of implementing advanced automation systems is the integration of newly developed technologies including AI planning and reasoning systems, advanced sensory perception, telepresence interfaces, obstacle avoidance and redundant arm control. Even though the PCR telerobotic system will require most of these advanced technologies, the program risk is not great. All the technologies will be demonstrated before this system is fully designed and implemented, with the exception of the redundant arm control.

1.1 Launch Pad Operations

A brief description of launch operations is required to understand the function of the PCR. The PCR is a self contained clean room located on the Rotating Service Structure (RSS) at the launch pad, see Figure 1. The RSS is attached to the Fixed Service Structure (FSS). The FSS provides access to the shuttle, SRBs and external tank. The RSS rotates up to the shuttle before launch, and is rotated away from the orbiter prior to lift off. The PCR provides access to payload bay and is the interface used to transfer vertical payloads to the orbiter.

Payloads are brought out to the pad in a vertical canister having the same dimensions as the orbiter payload bay. The canister is lifted off a transporter and raised approximately 80 feet, to the level of the PCR. Seals on the PCR provide an air tight bond between the canister and PCR. The doors to the PCR and cannister are then opened, and the payload is then transferred from the canister to the orbiter. The transfer is accomplished using the Payload Ground Handling Mechanism (PGHM) which lifts the payloads by their attached trunnions and retracts back into the PCR. The PGHM is actually a two degree-of-freedom device with lifting and translating capabilities. Once the payloads are secured on the PGHM, the doors to the cannister and PCR are closed, the cannister is lowered, and the RSS is rotated towards the orbiter. Once the PCR is against the orbiter, the doors of the PCR and orbiter are opened.

At this time the final preparations are made to payloads. The PGHM then translates out and lowers the payload into the orbiter bay. The trunnions attached to the payloads are secured in the orbiter fittings and final closeout testing is completed. After all processing is completed, closeout photos are taken and the doors to the orbiter and PCR are closed. The RSS is then rolled back and final launch preparations occur.

1.2 PCR Description and Processing Tasks

The PCR is an 80 foot tall Class IV (<10K 0.5 micron particles/foot3) clean room which provides limited access to the payloads. Platforms are fixed at various levels, approximately every ten feet in height. These platforms are extended as required towards the payloads in the orbiter. Often, these extended platforms do not provide the necessary access, so auxiliary platforms are attached to the original platforms with 'C' clamps, see Figure 2. These auxiliary platforms are commonly called 'diving boards'. These devices do not provide complete access to the payload interfaces in all instances due to their location. In addition, the diving boards do not provide safety rails and have limited load capabilities, which may expose the payload technicians to undesirable working circumstances.

The problem of access has been addressed in the past by erecting temporary scaffolding in the PCR and more recently by constructing special purpose access hardware in the PCR. The scaffolding has several advantages in that it is completely portable and reconfigurable. The major disadvantage is that during the erection of the scaffolding, the payloads are susceptible to damage. The sections of scaffolding are tethered to personnel who then climb and assemble the sections. This is a difficult task under any circumstance, but in the constrained environment of the PCR, this task becomes very challenging. By constructing special purpose access platforms, many of the difficulties associated with the original access platforms or additional scaffolding are eliminated. However the cost for these assemblies are high, and they are designed to be used only once or twice and then they are dismantled and scrapped. Recently, approximately $750K has been designated for the special access structures for the Magellan and Galileo missions. This cost is typical.

A practical alternative to the previous methods is a dedicated robot system within the PCR. A flexible manipulator system could provide much greater access to the payloads, be able to operate in highly constrained environments, and handle

hazardous materials. This robot would have to be specially designed to meet the many requirements of the PCR environment. Requirements include clean room operation, dexterous motion capability, and a high degree of safety and reliability.

2. BACKGROUND

Two specific driving forces have led to the highly positive consideration of an actual robotic system implementation at the PCR. First, the SS Strategic Plans and Programs Office-Advanced Development Program is sponsoring a joint JPL/KSC remote telerobotic demonstration program to integrate and advance a number of technologies that will be required for successful SS robotic applications. Secondly, there is a significant need for improvements in the current methods used for processing payloads at the shuttle launch pad. Improved, more flexible access to payloads is required to reduce the need for costly access platforms and fixtures, and eliminate the use of particulate generating cranes and lifting devices. A telerobotic demonstration program is currently being developed to meet the two tasks described above and will be described in this section.

2.1 Current Demonstration Program

In September 1988, JPL and KSC realized a common need and interest in a joint telerobotics program. JPL was interested in demonstrating remote teleoperation with induced time delays. KSC was interested in applying the technology to actual ground processing applications. By combining capabilities and resources of JPL and KSC, a leveraged program has evolved with an overall benefit to both current ground processing operations at KSC and meet the long range goals of on-orbit telerobotics for Space Station. The first year of a planned three year program is currently underway. A single application will be demonstrated this year.

Improvements in the software to meet the stringent requirements of the users in the PCR, hardware improvements to the user interface to provide real-time simulation of robot motions, the development of advanced proximity systems to improve the reliability of obstacle avoidance will be implemented in years 2 and 3. More advanced applications will also be demonstrated.

2.2 Demonstration System

The remote telerobotic test-bed will consist of three major components, a user interface and computer control hardware at JPL and a manipulator at KSC. Three leased 9600 buad serial communication lines will be used to connect the user interface and computer control hardware located at JPL to the robot located at KSC, see Figure 3. One line will be used to provide direct voice communication between the remote sites and the other two lines will be used to transmit compressed video images to JPL and provide two way robot communication, respectively.

The user interface at JPL resides in a Symbolics 3640 AI workstation. Using a menu driven, high level robot language, the operator is able to command the robot. The operator is able to preview robot motions with a graphic simulator on the Symbolics, observe joint positions and the natural language description of the current task being processed, see Figure 4. The Symbolics also does the task planning and reasoning and maintains the CAD model of the operational environment for obstacle free path planning.

The Symbolics machine is connected to a uVAX computer at JPL via a DECNET point-to-point connection. The Sensing and Perception module, and the Run-time Controller module are both located on the uVAX. The Sensing and Perception software is used to receive and reduce sensor information, and Run Time Controller outputs joint level robot commands. The uVAX at JPL is connected to the uVAX at KSC through a pair of leased serial lines. The uVAX at KSC is used to grab and compress images from a robot mounted video camera, and feedback joint positions from the robot and receive and execute robot motion commands via joint coordinates.

For direct teleoperation, a Symbolics computer will be installed at the KSC site, see Figure 5. The existing uVAX at KSC will then be directly connected to the Symbolics via a Decnet connection. The appropriate software will be copied from the computers located at JPL and loaded on the respective computers at KSC. The direct teleoperation scenario is preferable for testing the control software. Also for demonstration purposes, it would be more effective for the user/operator to see the response of the robot directly.

The manipulator located at KSC is an ASEA IRB-90, a six axis serial device with DC actuators. The ASEA IRB-90 has a 200 pound capacity, a 12 foot reach and a repeatability of 0.010 inch. It is located on a 30 foot track, and motion control along this track is provided by the robot controller.

2.3 Demonstration Applications

The first phase will demonstrate the capabilities of the user interface and controlling software. This will be done with the user interface being developed by JPL and ASEA IRB-90 robot at KSC. The potential users of the system (Shuttle Payload Operations) requested a realistic demonstration platform so that the telerobotics system could be realistically tested on actual flight hardware before being implemented in the PCR. The users recommended several pieces of demonstration hardware including a Payload Assist Module (PAM) Cradle, see Figure 6, and a Mission Peculiar Experiment Support Structure (MPESS). These two items were selected because of their geometry and constrained internal space. The users felt that if the robot could successfully navigate through such a constrained environment without hitting obstacles, then it could be easily used on less restrictive payloads, [3].

Approximately ten telerobotic applications have been identified by the users in the PCR and these include: component inspection, close-out photography, sharp edge inspection, lanyard identification and grasping, non-flight hardware identification, payload bay protective liner removal, insertion and removal of Quick Disconnects (QDs), and insertion of small items before closeout. These tasks are listed in order of increasing difficulty [3].

The first three tasks are strictly non-tactile tasks and require less sensory information and control software than the other tasks. The lanyard identification and grasping was included because it is a high priority item among the users and requires no direct contact with the rigid objects. The lanyards are attached to lens caps and other covers which must be removed before flight. Some of the lens caps and covers are currently designed for automation, and the others will be modified so that the robot may easily remove them by simply grasping the lanyard and pulling away from the payload [3].

The remaining applications are more difficult because they require some form of force feedback to the controller, or to the operator using force feedback joysticks. These tasks are currently scheduled to be accomplished in the second and third years of the program [3].

3. CRITICAL TECHNOLOGIES

A robot system located in the PCR must perform a wide range of tasks if it is to be cost effective, and well received by processing personnel. The highly constrained, delicate environment and difficult tasks necessitate the need for a system with a number of advanced capabilities. The PCR environment is vastly different than typical manufacturing applications. The required work area is highly cluttered and will contain extremely expensive, critical flight hardware and ground support equipment. Not only is the equipment critical, once it has reached the PCR a great deal of processing time has been expended and it is the last step in the payload launch flow. Thus, any damage would be extremely costly and significantly affect the launch schedule.

In order to implement a robotic system with the required capabilities a number of advanced technologies will be required. Considering the current capabilities of robotics and AI technology, the only way to accomplish the required tasks in a safe and reliable way is to provide a supervised, human augmented system. Human intelligence must be used to guide the planning process and react to uncertainties associated with unknown objects in the facility.

The most crucial elements necessary for the successful implementation of a robot in the PCR are a highly flexible, easy to use human interface which requires no previous robot programming knowledge, and fail safe methods to assure collision free motions of the robot. The critical technologies required to provide the needed capabilities include high level robot control languages, 3D object recognition and location vision systems, task scheduling, path planning, proximity sensing, collision avoidance, control of redundant manipulators, telepresence and force control. These technologies, with the exception of redundant manipulator control, have all been addressed by past research efforts and are currently being refined in the joint KSC/JPL PCR telerobotics demonstration program. Each of the required technologies are described briefly below.

3.1 Obstacle Avoidance Technologies

Obstacle avoidance must be guaranteed in a redundant fashion when working with sensitive flight hardware at the launch pad. Collisions of any links of the robot arm, end-effector, or tooling with any object in the work area must be

avoided under all circumstances. This can be accomplished using a high level controller which creates obstacle free path plans based on stored and real-time knowledge of the working area. In addition all motions and tasks will be graphically simulated and approved by human operators.

To insure maximum reliability, the system will incorporate a triple redundant obstacle avoidance system. Primary obstacle avoidance path planning will be performed by a control computer using CAD graphic models of all payloads, PCR, ground support equipment and the Shuttle bay. CAD models for the launch facilities and a majority of the payloads are available and will be stored in the control computer during each mission. In addition to the static data a 3D vision system will be used to identify the shape and location of unknown or moveable objects (antennas, valves, doors, holding fixtures etc.). This data will be used to update the models before a path is planned. In addition to these two systems an independent hardware based collision detection system will automatically shut down the system before an impending collision.

The obstacle avoidance path planning technology will be based on the techniques being implemented in the demonstration program (see Section 2.2). The path planning method developed at NASA-JPL, described in [4], is based on the free-space techniques originally developed by Lozano-Perez [5]. The technique has been modified however, by representing all objects in the workspace, as described by the stored geometric model, in terms of the manipulator joint coordinates. All paths are then planned in this joint coordinate representation which is referred to as the configuration space. The method is best described by the following procedural description:

1. The manipulator workspace is discretized into a set of p joint values for each of the N axes of the manipulator. This results in a table of N^P configurations.

2. The discretized configuration space is then searched to determine if any link or the end-effector interferes with any object in the workspace. If an interference occurs the node is marked as occluded space.

3. The above binary table is created and stored off-line. Online path planning consists of determining a path in the N-dimensional space between the current and desired location which contains only non-occluded nodes (free-space).

Generating the binary workspace obstacle map is a numerically intensive operation which takes considerable time. Thus this technique is only practical for relatively static work areas. The primary advantage of this method is that a collision free path for the entire arm is created. There are a number of other advantages of working in the joint coordinate representation which are discussed in [4].

Using the above technique which requires a completely static and known model of all objects accessible by the PCR robot is not adequate. The model and corresponding obstacle avoidance map must be updated due to moveable objects, tools and fixtures. This will be accomplished using a 3D object recognition and location vision system. A stereo vision system located on the robot will be used to recognize and determine the shape and location of known and arbitrary objects. This information will be used to update the CAD model of the work area used by the planning system. Image processing 3D recognition and location techniques are now in development at JPL [6] and a number of other laboratories. Although this is leading edge technology, based on past demonstrations at JPL, it is expected that a satisfactory system will be developed in the demonstration program.

One critical issue not addressed above is the availability of CAD graphic models of all objects in the PCR during processing. Models of the PCR are being developed now and a model of the Shuttle bay already exists. However, these models exist on various computer systems using different 3D graphic representation standards. Most of the payloads also exist in digital form. The critical technology therefore is the ability to transport a wide variety of CAD models into a common representation which can be accessed by the high-level computer. Current plans are to transfer all models into the Interim Graphic Exchange Specification (IGES) standard. However this may be inadequate for 3D models and advances in this technology may be required.

Hardware based collision detection technology is also required to avoid unexpected collisions and performance. A proximity sensing system capable of providing the distance between any point on the manipulator arm and the closest object of any material must be implemented. A large number of reliable proximity sensors must be mounted on the arm and integrated into a single, fast reacting system. Potential sensor candidates include sonar, radar, laser triangulation, coherent laser radar, and highly compliant contacts. This technology has not been demonstrated in large scale. However there is some direction in the Flight Telerobotic Servicer (FTS) program to provide a system of this type.

3.2 High Level Task Planning, Reasoning and Human Interface

A high level programming language and user interface will be required for the efficient operation of a robot in the PCR. This system should have the ability to plan, initiate and schedule complete operations based on generic input commands, and provide high resolution graphic simulations of all planned operations for review. The system should also be able to reason over a set of rules and guidelines which assures that all processes and tasks will be performed according to set procedures and Operational Maintenance Instructions (OMI).

A high level or generic input interface is required to assure that operators with little robot programming experience are able to quickly and easily program and operate the robot system. The high level programming system will primarily alleviate the need for tediously programming entire paths of end-effector positions and various tool commands. Using this interface the operator would teach or operate the system by selecting specific actions or operations from a menu to be performed on a given payload or piece of equipment, on a specific subsystem. For example, the operator would select 'INSPECT' from a menu (Inspect, View, Insert, Remove, Move To, Open etc.), then from another menu the device 'PAYLOAD' and from a sub menu 'FUEL UMBILICAL'. The task planner would then determine the complete sequence of events necessary to carry out the task based on procedures and guidelines stored in a data base, and would provide an obstacle free path for the entire robot.

This capability will be based on an advanced task planning system developed by JPL for their Demonstration Testbed Project [4] referred to as the Remote Mission Specialist (RMS). The RMS has two stages of plan generation, stage one is responsible for converting high level directives into a series of commands which tell what specifically needs to be done in the task space. This series of commands is then used as an input to the second stage of the planner, where they are converted to primitives which are executable by the robot controller. This planning operation is constrained by the procedures and guidelines which must be stored in a knowledge base. Considerable knowledge engineering effort will be required to transform the generic and mission specific PCR operations into a form suitable for the AI computer system. All generated paths may be previewed with the use of graphic simulation before they are carried out. Thus, graphic motion simulation technology will also be required.

3.3 Telepresence Man-Machine Interface

The high-level user interface technology described above may not provide complete flexibility. For highly complex or spontaneous tasks, it may be more effective to operate the robot in the traditional teleoperator mode, with the operator controlling the motion of the end-effector with a joystick interface. The joystick controller should be a highly transparent interface allowing the operator to control the end-effector with the natural motion of his hand. He should be provided with the "look and feel" as seen at the end-effector, thus the term "telepresence".

Telepresence will be accomplished by providing the operator with visual and force feedback. Force feedback can be supplied by measuring the end-effector forces with available force transducers, and applying the corresponding force to the operators hand using powered actuators on the joystick controller. Visual information can be provided by the vision system cameras and additional optional cameras. Using a number of cameras, located on the robot or fixed within the PCR, several views of the robot arm, end-effector and work area can be provided.

Natural, telepresent control cannot be provided by a directly coupled master/slave controller. Current state-of-art man-machine interface technology is able to provide the above capabilities by using a control computer to act as a flexible interface between the joystick and the robot. The joystick position or motion is interpreted and transformed into suitable, corresponding robot commands. At the same time, the force on the end-effector or tool is interpreted, and transformed into suitable commands to the joystick to apply corresponding forces. Thus a universal, flexible bilateral controller is achievable.

With this flexibility a highly capable interface is achieved by providing force and motion scaling and filtering for vibrations. More importantly the reference frame for motion or forces can be selected to match the end-effector frame, the current display frame etc. Additional features such as position or velocity control and the ability to re-reference the joystick position will also be required. The technology to provide these capabilities has been demonstrated at a number of laboratories [7]. A bilateral controller of this nature will allow the operator to perform difficult tasks such as QD removal and insertion and other tactile-like assembly operations. Complete autonomy of these more difficult tasks is not readily achievable in a practical system and thus human intelligence and sensory capability is required. A universal controller provides an ideal augmentation human capability and is essential.

3.4 Control of Kinematically Redundant Manipulators

When a payload is mounted within the PCR or Shuttle bay a highly constrained and cluttered work area exists. This severely limits the ability to avoid obstacles using a 6 degree-of-freedom (DOF, ie. the number of actuators in the system) manipulator arm. The available obstacle free work area of the arm will be significantly increased using a redundant system with 7 or more DOF. This is due to the fact that the most general 6 DOF robot has at most 16 possible configurations for a given position and orientation (pose) of the tool or end-effector. Note, general here refers to a completely arbitrary set of fixed geometric constants which include the angle between each pair of adjacent axes (twist angles) and the fixed distance between adjacent links (offset dimensions) of the arm. Furthermore, most standard industrial robots, which contain all parallel or perpendicular adjacent axes have either two of four possible configurations for a given pose. This limited number of configurations may not provide obstacle free access for a required pose.

Introducing an additional link for the robot (an extra degree-of-freedom), provides an infinite number of robot configurations for a given pose of the end-effector. The human arm, containing 7 DOF (with respect to motion constraint not actuation), is an excellent example of a redundant manipulator. This is evidenced by examining the case of the hand placed firmly on a table. Without moving the position or orientation of the hand the elbow can be placed in an infinite number of locations.

Unfortunately algorithms for controlling redundant manipulators are now in the developmental stage. Most of the research is aimed at optimising a given control parameter. Possibilities include minimizing energy, minimizing or balancing motor loads, maximizing the speed of motion, etc. These optimizations would be useful for on-orbit applications, where resources are limited. However, for ground operations these optimizations are not essential. The primary use of redundancy will be to provide obstacle free motion and increase the dexterity and available work area of the arm.

Thus, the key technology requirement is the development of obstacle avoidance and path planning techniques for the redundant system. Considerable advancement in this area may be required. However, certain techniques may naturally extend to the redundant case. For instance, the free path, configuration space technique described above (Section 3.1) can be implemented with a redundant system. The binary obstacle map represented in joint coordinates is simply a forward position analysis of the system which is easily accomplished regardless of the number of links in the system. However the binary map becomes a 7 (or more) dimensional space. This may become impractical to generate even offline, and search for free paths. Heuristic or rule based techniques may be necessary to provide manageable techniques which can be implemented with practical computer hardware. This is the single technology requirement in which existing capability may not be adequate.

4. SYSTEM DESCRIPTION

In this section the preliminary conceptual design and functionality of the PCR telerobotic system will be described. The system will consist of the following three major components: a manipulator capable of operation throughout the PCR, a hierarchical computer system and user interface, and sensor systems. The basic requirements of the system include the ability to perform inspection and other processing tasks. The system must not contaminate the Class IV clean room. Collision free motion must be guaranteed by a highly redundant, reliable obstacle avoidance system. Lastly, the system must be extremely easy to operate either in programmed or run-time control modes.

The manipulator will likely contain seven DOF to provide enhanced obstacle free motion capabilities. It will be mounted on a vertical rail attached to an existing structure in the PCR as shown in Figure 7. This will provide access to a majority of the payload area of the shuttle bay. The robot will provide approximately a 15 foot reach, 20 pound payload capacity and a positional accuracy of +/- 0.10 inch. A custom designed system with an optimized geometry for the required work area will be required. The arm will have to meet clean room requirements and all actuators must be explosion proof.

The end-effector will be designed to accommodate various tools, sensors and cameras. Quick connect tools may be used for some tasks. The video system may require special lens and filters, so they must also be designed for quick connect/disconnect. A multiple DOF articulated device may also be used to reach between the payloads and the bay.

The computer system will be a hierarchical, two layered architecture. The top high-level control computer will interact with the operator, and perform task planning, reasoning, programming and program storage and retrieval. The second layer contains the run-time control system. The high-level controller will most likely be an AI workstation, and the run-time controller will be a standard multi-processor computer environment. Various individual processors for sensors and end-effector systems will communicate with the run-time controller as well as the manipulator system. A joystick device will also be interfaced to this controller. A fully integrated system with all processing systems embedded within the controller would be ideal. However, this is unlikely to be possible. It may be possible to embed the manipulator servo controller since a custom system is being designed.

A majority of the computer processors will not have to be housed within the PCR. However, the operator workstation monitor, joystick controller and video displays should be located within the room for maximum viewing capability. All computer devices and displays located within the PCR will have to be industrialized systems to withstand the effects of launching.

The user interface consists of a joystick controller, video and simulation displays and the high-level controller workstation. The workstation will provide the primary interface to the system. The operator will have the capability of programming tasks by selecting generic descriptions of locations and devices from workstation menus. Taught or programmed tasks may be simulated graphically before actual execution. The work station will control the mode of the system (teleoperation, simulation, programmed task execution etc.) and provide required status. A force-feedback joystick controller will likely be required to perform assembly tasks. Thus the system will be capable of running in supervised teleoperation mode. Supervised meaning the obstacle avoidance system will continue to run in this mode.

A number of sensor systems of various complexity will be required. The obstacle avoidance system is based on two individual sensor systems as explained in Section 3. A 3D video image processing system will be required to recognize and determine the location and orientation of arbitrary objects. Also, the coordinates of carefully designed reference targets throughout the area will have to be determined. An arm based proximity sensing system will also be required to warn of impending collisions of any point on the arm. A hardware based system integrating a large number of small sensors mounted on the arm is envisioned. Each proximity sensor will be required to determine if a minimum distance, along a straight line, to the closest object of any material has been reached. Standard force/torque sensors will also be mounted on the end-effector.

5. DESIGN ISSUES

A number of major design issues will have to be addressed for the implementation of a telerobot system at the launch pad. The key issues include the following areas: arm geometry, system mounting and mobility, clean room requirements and, computer architecture and partitioning.

The kinematic structure or arm geometry (the fixed geometric parameters) of the robot must be designed to provide adequate access and dexterity, and allow obstacle free motion for number of applications and payloads. A six DOF arm could be designed to provide all desired positions and orientations of the end-effector in an uncluttered environment. However, in the highly constrained environment of the PCR, this may be impossible. As explained in Section 3.4, adding DOF to the robot provides an infinite number of configurations for a given end-effector pose. A detailed study using robot system workcell simulation and analysis tools will be required. Models of various payloads, the payload bay and the PCR will be used to determine the capabilities of various robot designs. A redundant system will only be implemented if a six DOF system cannot be designed to meet a majority of the desired tasks.

Because of the large dimensions of the payload bay, it would be impractical to design a stationary robot to provide the required access. Therefore the robot must be either allowed to move on vertical tracks or be easily transported to various locations within the PCR. To provide portability for the robot, it would have to be disassembled, re-assembled and calibrated before being used on the actual flight hardware. This greatly reduces the flexible capabilities of the system.

At this time, it appears that attaching vertical tracks to the PCR would provide the most effective coverage of the robot. However, the major drawback of tracks is the potential of it generating a large number of particulate contaminants because of the interaction of wheels against the track. This problem may be circumvented by placing protective covers over the track and the wheels.

Proximity sensors will be attached to the entire robot so that all links are instrumented for obstacle avoidance. The robot will incorporate explosion proof DC actuators with internal encoders, tachometers and fail safe brakes. The arm will also be designed to operate in a Class IV clean room. The above requirement makes actuator and drive system design a formidable task. All drive systems will have to be enclosed or gearless and may have to use dry lubrication to reduce contaminate generation. Possibilities include the use of harmonic drives, high-capacity step motors, direct drive motors, enclosed mechanisms or specialized motors.

The two layered computer architecture concept described in the above section will be adhered to. However, a number of specific design issues must be addressed. The two main computer systems will most likely communicate over a networked connection. The specific requirements and throughput rates of this connection must be established. The critical design issue then become the selection of a suitable network protocol. A uniform, easily maintained method of communicating and interfacing the various stand-alone processor systems to the run-time controller must be established. This is the only way to provide a reliable and flexible system capable of being expanded in a practical manner. Optimal selection of a suitable, standard bus structure (VME, MultiBus II, NUbus etc.) for the run-time controller, which will contain

a large number of processors, is also a critical issue. Because the of the location of the PCR and its close proximity to the shuttle during lift-off, all equipment will have to be designed to withstand the associated shock, vibration and heat.

6. POTENTIAL APPLICATIONS

A multitude of tasks in the PCR are potential candidates for automation. These tasks include component inspection and verification, close-out photography, non-flight hardware identification, payload bay protective liner removal, lens cap removal, insertion and removal of QDs (quick disconnects), and insertion of small flight batteries and film packs.

These tasks can be divided into two categories, those tasks which do not require the robot end-effector to touch flight hardware, and those tasks which do. The first group of tasks eliminate all of the difficulties associated with force control. As long as the obstacle avoidance software works properly, there should be no physical interaction between the payload and the robot. This greatly reduces the risk factor associated with operating a robot in the PCR. However, many of the tasks which require contact provide the highest pay back.

Three tasks are of the highest priority, inspection of payload components for sharp edges, lens and dust cap removal, and insertion and removal of QDs. The first task is a non-contact in which astronauts personally check all payloads in the payload bay for sharp edges. Sharp edges could cause space suits to tear and depressurize. Currently, this requires special scaffolding to be erected so that the astronauts may thoroughly inspect the entire payload bay before close-out. A camera would be mounted on the end-effector, and the robot could either be controlled manually with the joystick, or automatically with the high level control software. Images of the payloads taken from the camera would be transmitted to a monitor and recorded for future reference.

On many payloads, a number of lens caps and dust covers are used to protect optical and other surfaces from contamination. Usually, these are the last non-flight items removed from the payload bay before closeout. Often, these protective covers are located in places with limited or non-existent access. In the past, technicians have walked on flight hardware to reach these locations and while nothing was damaged, the potential for damage and flight delay is great. The lens caps and dust covers must be designed for automation in the future. Presently, lanyards are attached to the lens caps and dust covers so that a technician can easily remove them by pulling on the lanyard. A visual target could be attached to the lanyard, and the integral vision system could identify and direct the robot to the target. Because the lanyard is compliant and will not transmit forces towards the payload, this task does not require extensive force control capabilities.

The third major task is the most difficult. Connecting and disconnecting QDs requires extensive force control capabilities coupled with object identification, positioning, and path planning. Currently, the Robotic Applications Development Laboratory (RADL) at KSC is involved in automated QD insertion for remote umbilical connections. This research has demonstrated successful target acquisition and insertion of a QD into a receptacle. The QDs include fluid, gas, power, and communication connections. The QDs are located about the complete periphery of the payload, and are often located in inaccessible locations. For example, the upcoming Magellan has over 50 QDs in various locations. QDs designed for automation greatly simplify the required robotic capabilities. For example tapered shanks, self aligning and automatic locking QDs which incorporate common design for different missions will improve the robot capabilities and help to reduce processing costs.

7. CONCLUSION

Payload and shuttle processing tasks within the PCR represent an ideal opportunity for improvements through the use of physical automation and telerobotic technology. A reliable, easy to use manipulator system capable of providing access to a large portion of typical payloads within the shuttle bay will reduce processing costs and potential contamination, and improve safety and cleanliness. Additionally, the implemented technologies and system designs will also provide similar benefits to both on-orbit and ground processing of the Space Station Freedom. A number of advanced technologies will have to be integrated in the proposed system. However, these technologies have been developed and are currently undergoing refinements in full scale demonstration programs, greatly reducing the associated risks.

The advanced technologies and capabilities required for the proposed telerobotic system include a redundant obstacle avoidance system, intelligent task planning and reasoning, high level user programming interface, force feedback joystick control and potentially, redundant arm control. This system represents an optimal balance between system autonomy and human intervention based on todays technical capability. Inherent in this augmented or balanced system design is the ability to evolve to a higher degree of autonomy. An initial implementation capable of performing simple placement and scanning tasks, without joystick control capability can be implemented in a 2-3 year period. A complete

implementation is possible within 5 years. The completed system will represent effective and rapid use of NASA developed, state-of-the-art automation technology.

REFERENCES

1. Space Station Processing Facility Automation and Robotics Study, WO-P0005, McDonnell Douglas Astronautics Company, Kennedy Space Center, Florida, 1987.

2. Gary Sweet, Leonard Hutto, Robots Refurbish Space Shuttle Hardware, SME-RI Robots 11 Proceedings, 1987, p 11:9-28.

3. Dan Wegerif, Mike Sklar, Plan For JPL/KSC Telerobot PCR Demonstration, McDonnell Douglas Astronautics Company, Kennedy Space Center, Florida, 1988.

4. Steve Peters, Carol Collins, David Mittman, Jacquelyn O'Meara, Mark Rokey, Planning and Reasoning in the JPL Telerobot Testbed, Jet propulsion Laboratory, California Institute of Technology, Pasedena, California, 1988.

5. Tomas Lazano-Perez, Spatial Planning: A Configuration Space Approach, IEEE Transactions on Computers, Vol. c-32, No. 2, February 1983, pp. 108-120.

6. Brian Wilcox, D. Gennery, B. Bon, T. Litwin, The Sensing and Perception Subsystem of the NASA Research Telerobot, Proceedings of the Workshop on Space Telerobotics, JPL publication 87-13, p 3-8.

7. D. Tesar, H. Lipkin, Assessment for the Man-Machine Interface Between the Human Operator and the Robotic Manipulator, (NSF Grant No. ENG78-20112 and DOE Contract No. ER-78-S-05-6102), Univ. of Florida, 1979.

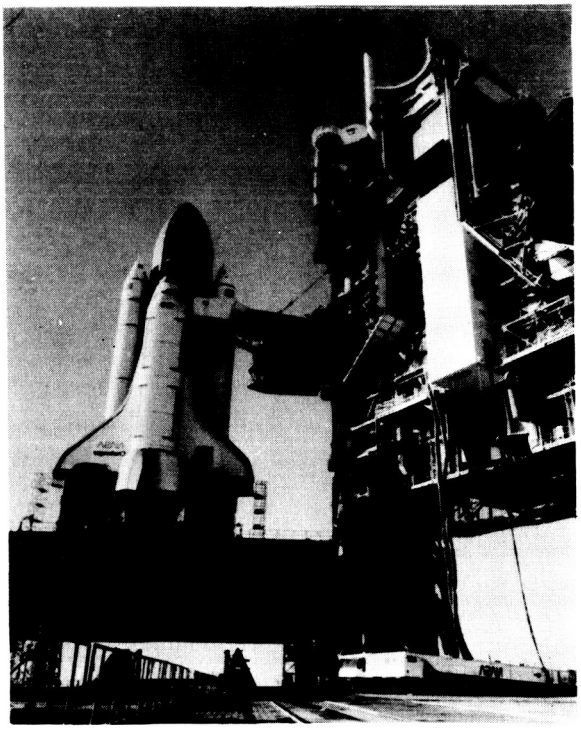


Figure 1. STS, RSS, Payload Canister And PCR

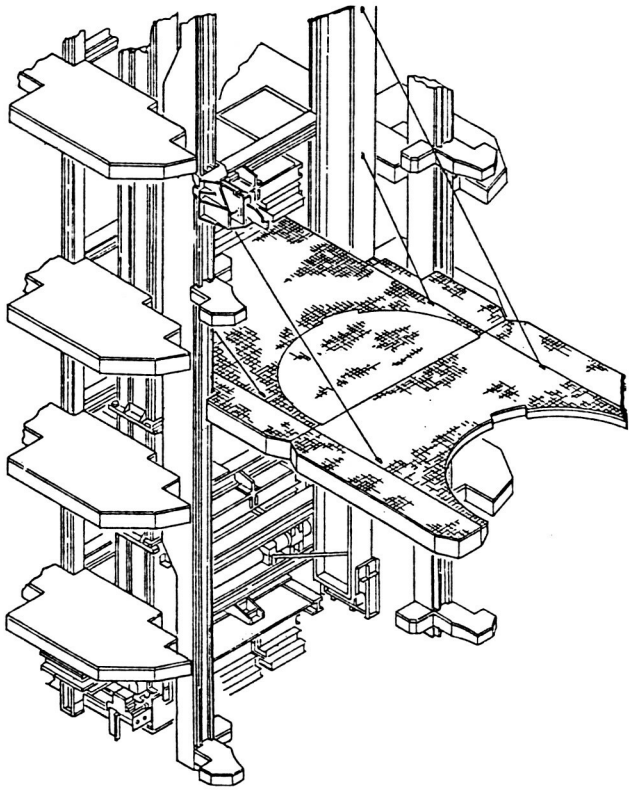


Figure 2. Payload Changeout Room

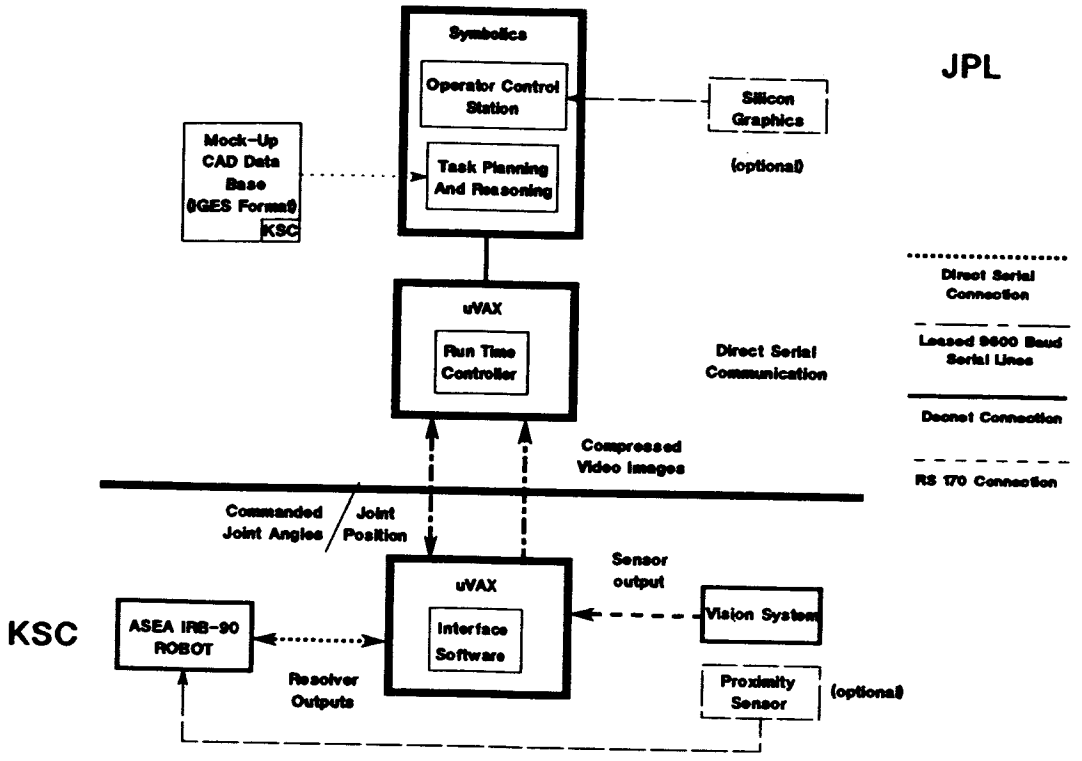


Figure 3. KSC/JPL Hardware and Communications For Remote Operations

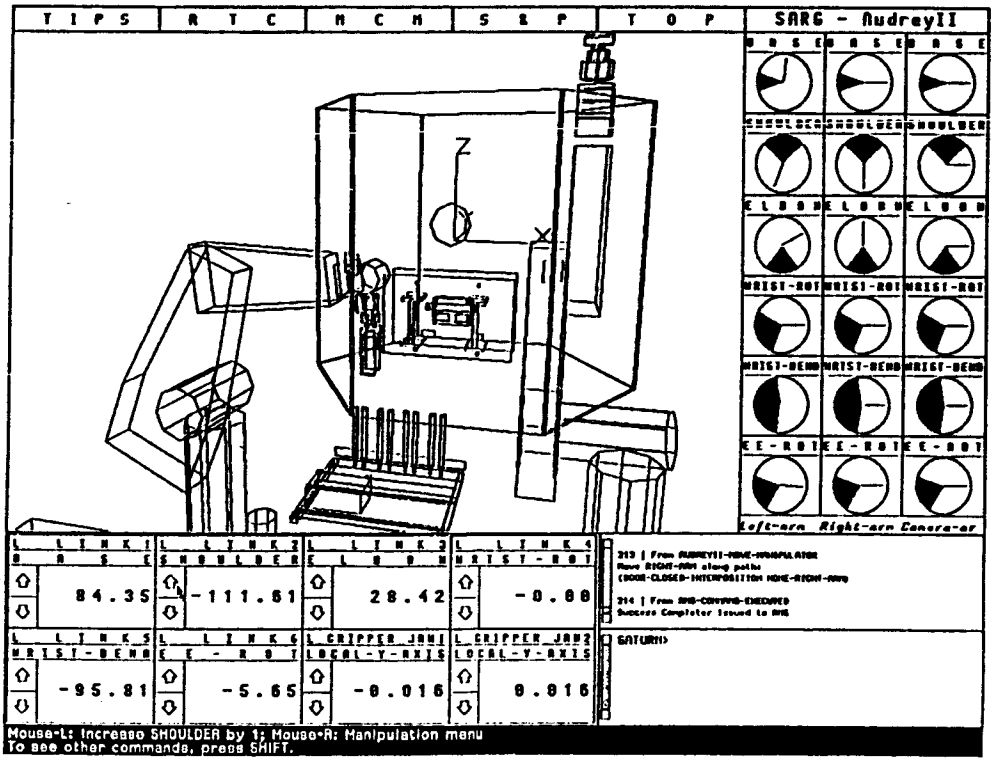


Figure 4. Current JPL Teleoperator User Interface

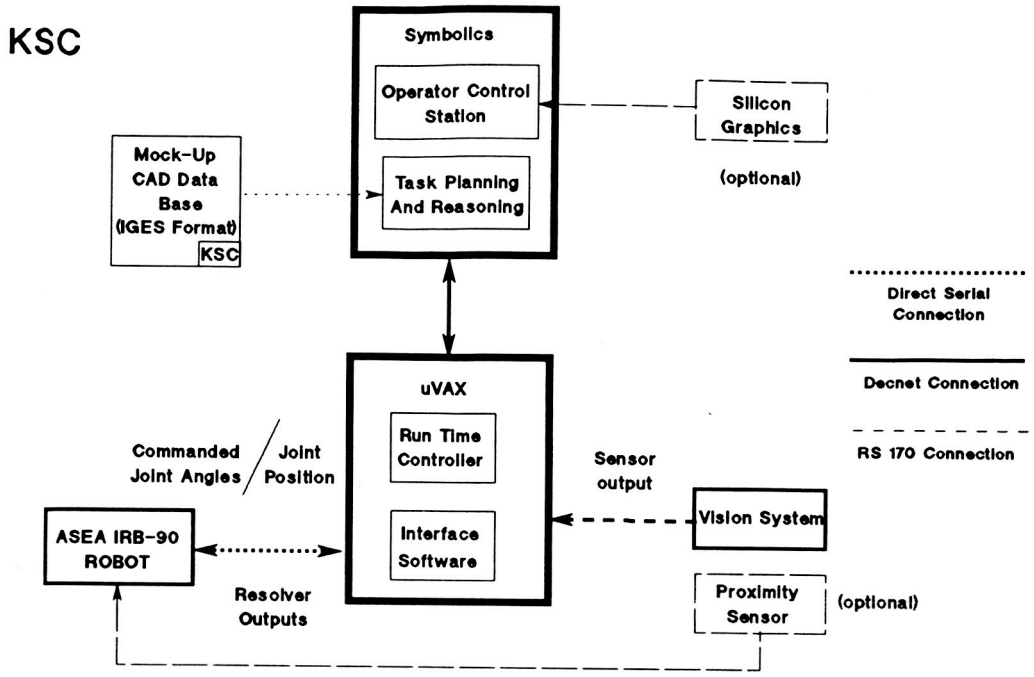


Figure 5. KSC/JPL Hardware and Communications for Direct Operations

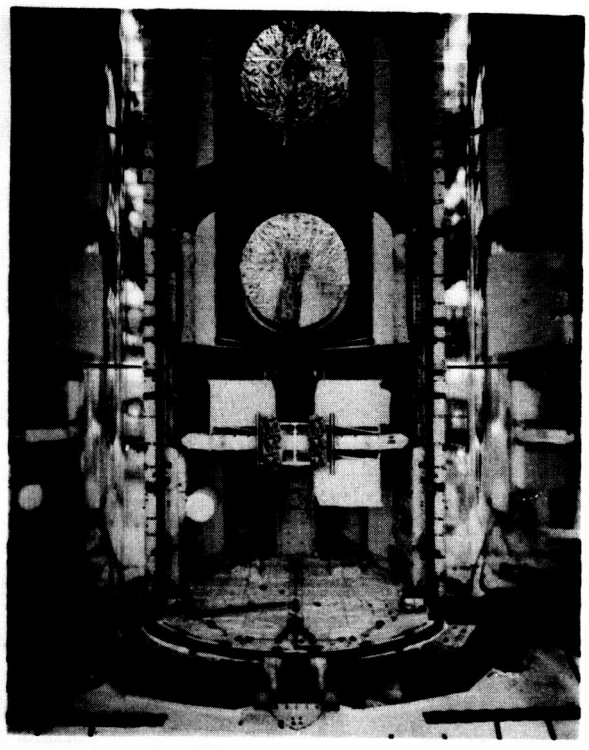


Figure 6. PAM-D Payloads Mounted In STS Payload Bay

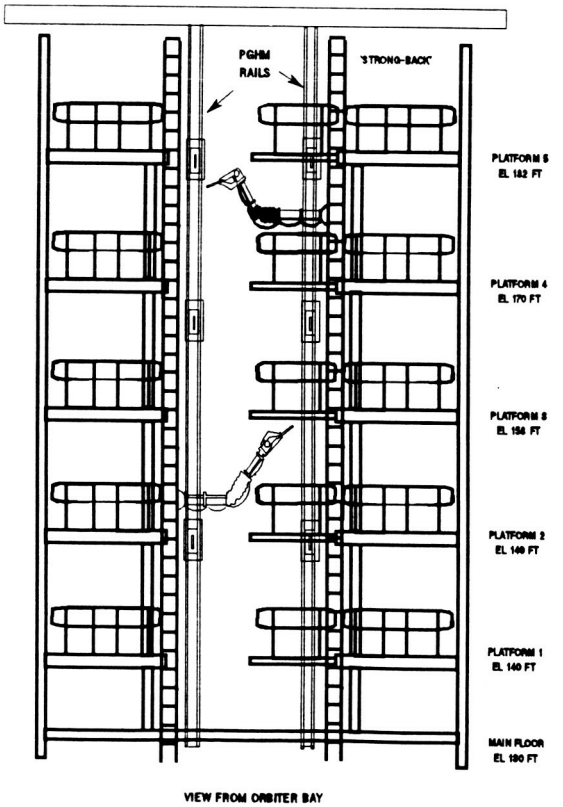


Figure 7. Conceptual Design of Manipulator Located In The PCR

ROBOTIC TELE-EXISTENCE

Susumu Tachi, Hirohiko Arai and Taro Maeda

Mechanical Engineering Laboratory, MITI
1-2, Namiki, Tsukuba 305 JAPAN

Abstract

Tele-existence is an advanced type of teleoperation system that enables a human operator at the controls to perform remote manipulation tasks dexterously with the feeling that he or she exists in the remote anthropomorphic robot in the remote environment. In this report the concept of the tele-existence is shown, the principle of the tele-existence display method is explained, some of the prototype systems are described, and its space application is discussed.

1. Introduction

Tele-existence aims at a natural and efficient remote control of robots by providing an operator a real time sensation of presence. It is an advanced type of teleoperation system that enables a human operator at the controls to perform remote manipulation tasks dexterously with the feeling that he or she exists in one of the remote anthropomorphic robots in the several remote environments [1,2]. Similar concept is called artificial reality [3] or telepresence [4].

Fundamental studies for the realization of the tele-existence system are now being conducted in the authors' division of the Mechanical Engineering Laboratory (MEL) as part of the National Large Scale Project called JUPITER (JUvenescent PIoneeing TEchnology for Robots), which is a research and development program of advanced robot technology for a system that avoids the need for humans to work in potentially hazardous working environments, such as nuclear power plants, under sea, and disaster areas.

In previous papers [1,2], the principle of the tele-existence display method was proposed. Its design procedure was explicitly defined. Experimental display hardware was made, and the feasibility of the visual display with a sensation of presence was demonstrated by psychophysical experiments using the test hardware. In the latest paper [5], a method was proposed to realize a mobile tele-existence system, which can be remotely driven with the auditory and visual sensation of presence. A prototype system was constructed and the feasibility of the method was evaluated. The effectiveness of the proposed system was evaluated by navigation experiments of the mobile robot through an obstructed space. Several display and operation methods were compared quantitatively using the time elapsed, smoothness of the travelled path and the number of collisions as the criteria for comparison.

In this report the concept of the tele-existence is shown, the principle of the tele-existence display method is explained, some of the prototype systems are described including an anthropomorphic robot with seven degrees of freedom arm, which is designed and developed as a slave robot for feasibility experiments of teleoperation using tele-existence method.

2. Tele-existence

In tele-existence, an autonomous anthropomorphic robot is placed at a remote site and an information transmission communication channel is established between human and robot. The operator's movements and physical status are sensed and transmitted to the robot via this communication channel. The transmitted signals override the autonomy of the robot and directly control the robot's motor system to reproduce the exact movements of the operator in its artificial eyes, neck, hands, legs, and feet. Information picked up by the artificial sensory organs of the robot are transmitted back to the operator via the communication channel to the operator's sensory organs.

Take vision for example. Whichever direction the operator looks, the robot will look in the exact same spot. The operator will see on his retinae the image seen by the robot, in exactly the same manner as it would be seen by a human in the same position. If the operator were to bring his arm in front of his eyes, he would see the robot's arm being brought into his field of view in exactly the same relative position as his own arm. Thus the operator is able to maintain his or her visual sensation and proprioceptive sensation coherent. The operator can perform tasks via a robot at a distance yet maintain the same spacial relation among the objects, the arms and the environment as that by direct observation. Auditory and tactile sensations are also transmitted to the operator. Objects touched by the robot are also felt by the operator as tactile stimuli.

Tele-existence technology also goes beyond the scope of the human senses. Radiation, ultra-violet rays, infrared, micro waves, ultrasonic waves, and ultra low frequency sound information sensed by the robot sensors can also be utilized to augment the human operator. For example infrared information picked up by the robot sensors can be converted into visible light on the operator's display. As the display gives a realistic sensation of presence, tasks can be performed in the dark yet with the illusion that it is light. These pieces of information can also be superimposed on the visual display as three dimentional superimposition. For example, by adding distance information at the location of an object. It is also possible to display human like solid model arms instead of the mechanical robot's arms, which enhances the operator's sensation of presence.

The final version of the tele-existence system will be consisted of intelligent mobile robots, their supervisory subsystem, a remote-presence subsystem and a sensory augmentation subsystem, which allows an operator to use robot's ultrasonic, infrared and other, otherwise invisible, sensory information with the computorgraphics-generated pseudorealistic sensation of presence. In the remote-presence subsystem realistic visual, auditory,

tactile, kinesthetic and vibratory displays must be realized [1].

3. Principle of Tele-Existence and Means of Realization

The basic configuration of the tele-existence system is shown in Fig. 1. Take vision as an example to explain the principle of the display which gives a sensation of presence [1].

The system is based on the principle that the world we see is reconstructed by the human brain using only two real time images on the two retinae of a human. What we get from the environment are only two-dimensional pictures on the retina changing in real time according to the movement of the eyeballs and the head. We reconstruct the three-dimensional world in the brain and project the reconstructed world to the real three-dimensional world [1].

In our new type of robotic display;
(a) human movements including a head and/or eyeballs are precisely measured in real time,
(b) robot sensors and effectors are constructed anthropomorphically in function and size,
(c) movements of the robot sensors are controlled precisely to follow the human operator's movement, and
(d) the pictures taken by the robot sensors are displayed directly to the human eyes in a manner which assures the the same visual space as is observed directly at the robot's location.

This display enables an operator to see the robot's upper extremities, which are contrlled to track in real time precisely the same movement of the operator's, instead of his/hers at the position his/her upper extremities should be.

4. Design of the Visual Display with Sensation of Presence

Essential parameters for human three dimensional perception of an object are: (1) the size of the retinal image of the object, or visual angle, (2) convergence of the two eyes, or equivalent disparity of the two retinal images, and (3) accomodation of the crystalline lenses. Adding to the above monochromatic parameters, fidelity in color is important for a realistic display [5].

Figure 2 (a) shows a schematic diagram of the direct observation of an object in three dimensional space. The human observer measures the convergence angle (α) and the size of the object on the retina (lm). Since the distance between the two eyes (Wm) and the distance between the crystalline lens and the retina (am) are known, a human observer can estimate the distance to the object (dobj) and the size of the object (lobj) as follow [5]:

$$dobj = Wm/2\tan(\alpha/2) \quad (1a)$$
$$lobj = dobj*lm/am \quad (1b)$$

If we think of a virtual plane at a distance of dvir perpendicular to the direction of the head; and project the object image onto the plane as shown in Fig. 5, and the human observer observes the projected images by

using the corresponding eyes, then the observed parameters, i.e., α and lobj, are the same and the human observer gets the same lobj and dobj. The lobj and dobj can be derived by using the equivalent disparity (ed) on the virtual plane and the projected image size on the virtual plane (lvir) as follows:

$$dobj = Wm*dvir/(Wm-ed) \quad (2a)$$
$$lobj = dobj*lvir/dvir \quad (2b)$$

where dvir is the distance to the virtual plane.

Figure 2(b) shows the display system which reproduces the same situation as the direct observation. Two TV displays and lens systems produce the virtual images of the size lvir on the virtual plane at the distance of dvir with the equivalent disparity of ed.

Figure 2(c) shows the slave robot's camera system, where the distance between lenses Ws is set to be equal to Wm. The distance between two CCD devices (wcam) is usually ,but not necessarily, set as Wcam=Wdis=Ws, where Wdis is the distance between the two centers of the TV displays.

Under these conditions, we define a magnification factor γ = ldis/ls. Then by arranging am=γ*as, we have the condition of Fig.6, which is the same condition as for a direct observation. Practically, am can be determined by measuring the size of the image on the diplay (ldis) when monitored through the TV camera for a known size object lobj at the known distance dobj as: am = β*dobj, where β = ldis/lobj.

The focal length of the lens (fm) must be selected to meet the condition that the virtual image of the TV display is on the virtual plane.

Ideally the distance to the virtual plane (dvir) should be controlled to coincide with the dobj controlling both fm and am. However, experiments revealed that if 200 mm $\leqq$ dobj $< \infty$, dvir can be fixed to 1000 mm, and if 145 mm $\leqq$ dobj $\leqq$ 2000mm, dvir can be fixed to 500 mm. This makes the design and realization of the system more practical.

If these conditions are satisfied and the cameras and the display system follow the head movement of the operator, the ideal condition of the direct observation is always maintained [5]. In order to have a wide view without moving the operator's head, a short focal length of the camera (fs) must be selected and the appropriate values for as and am must be set.

5. Design and Control of Display Mechanisms

5.1. Master-Slave Controlled Active Display Mechanism [2]

Figure 3 shows the experimental hardware system for the evaluation study. The movement of the head of the human subject is measured in real time by the light weight goniometer with six degrees of freedom. Three translational coordinates (x,y,z) and three rotational angles (roll,pitch, yaw) are calculated by a microprocessor, and both the camera position/

orientation and the display position/orientation are servo-controlled to follow the head movement.

As the number of degrees of freedom of the allowed head movement is large, it is impossible to use the torque produced by the operator's head movement as the energy source of the movement of the display device. Even if the weight is removed by a counter balance mechanism, the inertia can not be eliminated. Therefore, it is necessary to servo-control the display device.

The active display mechanism shown in Fig. 3 has five degrees of freedom. Each degree of freedom is actuated by a direct drive torque motor (Inland Rare Earth D.D.Torque Motor). The dispaly follows the goniometer's movement like a master-slave manipulator system with the goniometer as a master and the display as a slave.

Visual displays were designed according to the procedure proposed in section 4 using two 1.5 inch color CRTs. The visual tele-existence system with these displays were experienced by several subjects. All subjects had an impression that this type of display produces very realistic feeling of remote presence. Adding to the qualitative evaluation, objective and quntitative experiments were also conducted [2].

Experiments with this hardware revealed, however, that the position/ orientation control is not enough. Subjects usually want a compliant motion which follows their head movement. Therefore, force control based on the measurement of the head movement and/or force condition becomes necessary.

5.2. Impedance Controlled Active Display Mechanism

Figure 4 shows a general view of the impedance controlled head-coupled display with two degrees of freedom. It has an active power assistance mechanism and its impedance can be controlled by internal feedback loop. We used direct drive motors to attain this mechanisms, and the dedicated computer controls the impedance of the display mechanisms so that the human operator feels only quite low inertia compared with the physical inertia of the system.

The dynamic equation of a system with one degree of freedom can be expressed as follows:

$$KtI + To = J\theta s^2 + Fb\theta s + Fc, \qquad (3)$$

where θ is the motor rotary angle, I is the motor current, Kt is the sensitivity of the motor torque, To is the torque caused by the manual force, J is the moment of inertia, Fb is the viscous friction coefficient, and Fc is Coulomb's friction torque.

By substituting

$$I = (\alpha J\theta s^2 + \beta Fb\theta s + \gamma Fc)/Kt, \quad (0 < \alpha, \beta, \gamma < 1) \qquad (4)$$

into equation (3), we find that

$$To = (1-\alpha)J\theta s^2 + (1-\beta)Fb\theta s + (1-\gamma)Fc, \qquad (5)$$

which exhibits the effects attainable by multiplying the inertia force, viscous friction force, and Coulomb's friction force $(1-\alpha)$, $(1-\beta)$, and $(1-\gamma)$ times as high, respectively. In other words, it is possible to redesign the motion equations of the system (or impedance of the system) into an arbitrary form through internal feedback [6,7]. An extention of the method to the multiple degrees of freedom system is shown in [6].

Figure 4(a) shows the impedance controlled display using two 3 inch color LCDs, and Fig. 4(b) shows the anthropomorphically arranged slave camera system with two degrees of freedom. When operators actually wore the display and moved it by neck force, the reaction force caused by inertia appeared to be lighter, and they reported that the difference was particularly noticeable when the display was moved swiftly. Operators felt that the system is quite similar to a passive mechanism of lighter weight.

5.3. Head Mounted Display

A Head mounted display is also a promising design approach. The merit of the head mounted display is that an operator can move around quite easily, while that the human operator must support all the weight by himself becomes its demerit. Since gravitational force and the inertia of the system can not be compensated in this system, the design of light weight display is quite important.

Figure 5 shows the head-mounted display Mk. I. It weighs 1.7 Kg, including a helmet (620 g for the display). It uses two 4 inch color liquid crystal TV displays (resolution: H320 x V220). Eye lenses which are used to attain the effect of Fig. 2(b) are mounted on a spectacles' frame. Lighter version of the head mounted display Mk. II has been made. Its total weight is 600 g.

6. Tele-Existence Experimental Systems

6.1. Mobile Tele-Existence System [5]

A prototype system with fundamental mobile tele-existence functions has been assembled for experimentation. The system consists of an independent mobile robot with two TV color cameras, a remote control station with the visual and auditory displays with a sensation of presence, and a communication link between the human operator and the mobile robot.

During routine navigation tasks, the robot travels autonomously using the environmental map and the environmental information gathered by the visual sensors (two TV cameras and an ultrasonic sensor) and internal sensors (two odometers on the rear wheels).

The navigation process can be monitored by the operator. When the robot encounters a task which the robot is not able to manage by itself, it stops and asks the operator for help. At that time the operator controls the robot using joysticks as though he were driving that robot like an automobile, i.e., as if he were on board the robot at the position where the robot's TV cameras are located.

Figure 6(a) shows the head-linked display with a sensation of presence used in the system, while Fig. 6(b) shows the prototype tele-existence mobile robot constructed.

6.2. Tele-existence Manipulation System

Figure 7 shows a general view of the anthropomorphic slave robot with an operator wearing head-mounted display explained in Section 5.3. Figure 8 shows an operator using the impedance controlled display explained in Section 5.2. In the latter system electromagnetic sensor is used for the control of the manipulator.

The slave robot has a three degrees of freedom neck mechanism on which stereo camera is mounted. The robot's structural dimensions are set very close to those of human's, and it is controlled to follow the human movement.

The human operator's head movement is measured in real time using electromagnetic sensor and the slave robot's neck is controlled to follow the master's movement. The streo color video signals are sent to the human operator and displayed as a fused image, which keeps the distance, direction and size of the object as those of the direct observation.

7. Tele-Existence Simulator

Extension of the tele-existence to the artificially constructed environmental information has been sought, the visual tele-existence simulator has designed, pseudo-real-time binocular solid model robot simulator has been made, and its feasibility has been experimentally evaluated [8].

Two main situations for the simulator usages are:
(1) To provide the operator information of the remote environment which human senses do not work but the robot's sensors do. For example, at night infrared sensor information is converted to visible light to see an object in the dark. It is also possible to superimpose range information gathered by the robot's ultrasonic and/or laser range sensors to the three dimensional visual display. The operator can effectively use this piece of information to augment human ability.
(2) To provide totally artificial but realistic environmental information to the operator, e.g., realization of virtual terminal or virtual console for the operator [3]. The operator can enjoy variety of consoles without changing them physically. This can also be used for the simulation study for training and also for optimal parameter selection and evaluation of man-machine system. The usage of the system as scientific tools for the analysis of human visual sensation, motion control and sensor-motor coordination is also possible.

As the first step toward the goal, a solid model robot manipulator with pseudo-real-time shading capability was constructed. By using the specially designed binocular optical system, three dimensional observation, which can exactly assign the distance and the size of the manipulator and an object, became possible.

8. Space Applications

Wide variety of application of tele-existence technology to space can be conceived, including tele-observation, tele-maintenance, tele-construction, tele-experiment, and tele-experience.

In Japan the Space Robot Resarch Planning Committee for the Ministry of International Trade and Industry has been organized. In the committee a space robot which replace some of the human extravehicular activities is being planned. Tele-existence will be intensively applied for the design of the space tele-robot.

In order to apply tele-existence technology to a tele-robotic system whose slave robot is located far away and the time delay for communication can not be neglected, it is important for a slave robot to be autonomous. Foundamental study for the realization of tele-existence using autonomous robots as slaves under the circumstances is now being conducted in the authors' laboratory.

References

[1] S.Tachi et al.,"Tele-existence (I) -Design and evaluation of a visual display with sensation of presence-," Proceedings of the 5th Symposium on Theory and Practice of Robots and Manipulators (RoManSy 84), pp.245-254, CISM-IFToMM, Udine, Italy, June 1984.

[2] S.Tachi and H.Arai,"Study on tele-existence (II) -Three dimensional color display with sensation of presence-, Proceedings of the '85 International Conference on Advanced Robotics (ICAR), pp.345-352, Tokyo, Japan, Sept. 1985.

[3] S.S.Fisher et al.,"Virtual environment display system," ACM 1986 Workshop on Interactive 3D Graphics, pp.1-11, Chapel Hill, North Carolina, October 1986.

[4] J.D.Hightower, E.H.Spain et al.,"Telepresence: A hybrid approach to high-performance robots," Proceedings of the '87 International Conference on Advanced Robotics (ICAR), pp.563-573, Versailles, France, October 1987.

[5] S.Tachi, H.Arai, I.Morimoto and G.Seet,"Feasibility experiments on a mobile tele-existence system," The International Symposium and Exposition on Robots (19th ISIR), Sydney, Australia, November 1988.

[6] H.Arai and S.Tachi,"Force detection and active power assistance of a direct-drive manipulator," Advanced Robotics, vol.2, no.3, pp.241-257, 1987.

[7] H.Arai and S.Tachi,"Development of a power-assisted head-coupled display system using a direct-drive motor," Advanced Robotics, vol.3, 1989 (to be published).

[8] S.Tachi, H.Arai and T.Maeda,"Tele-existence simulator with artificial reality (1), -Design and evaluation of a binocular visual display using solid models-," Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, pp.719-724, 1988.

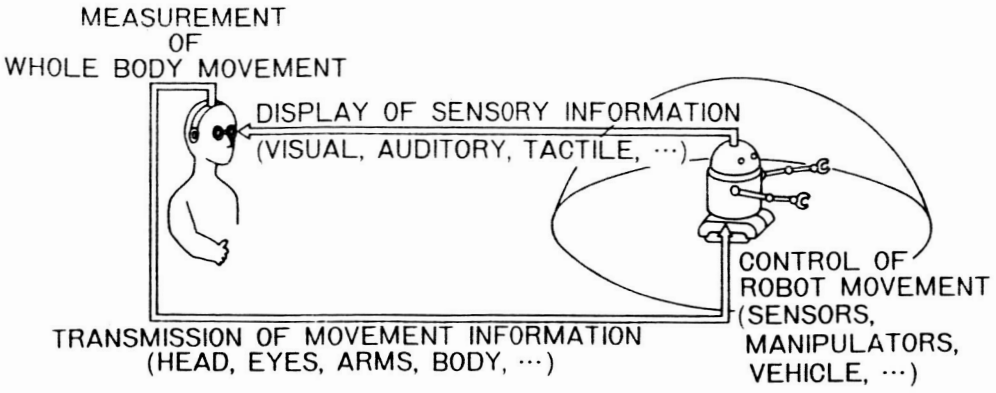
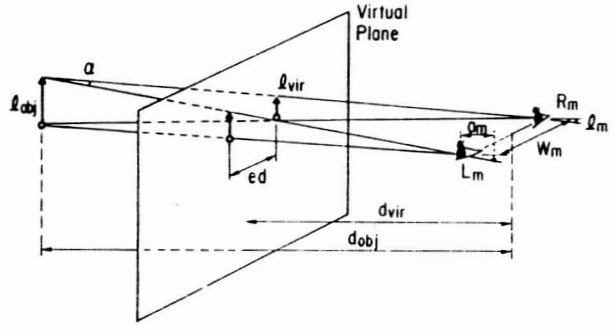
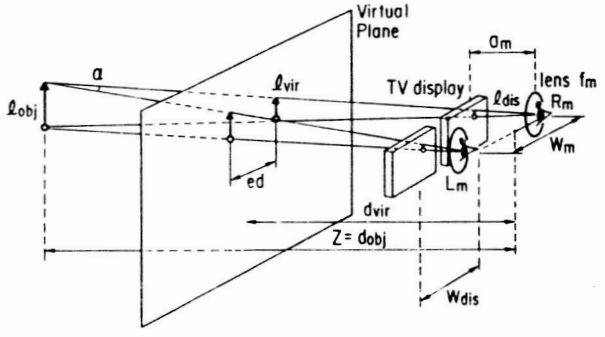


Fig. 1 Concept of the tele-existence.



α : convergence angle
ed : equivalent disparity

Fig. 2(a) Visual parameters of direct observation.



α : convergence angle
ed : equivalent disparity

Fig. 2(b) Parameters through the display.

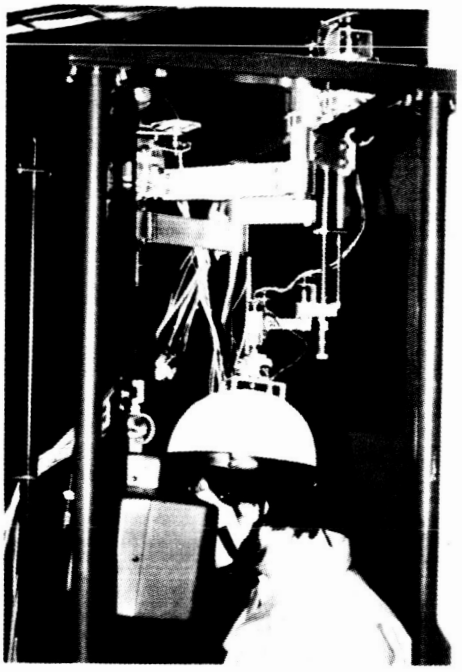


Fig. 3 Master-slave controlled active display.

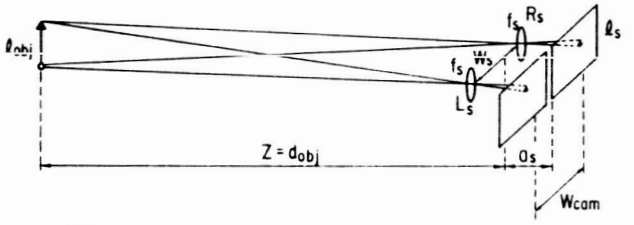


Fig. 2(c) Parameters of the slave robot.

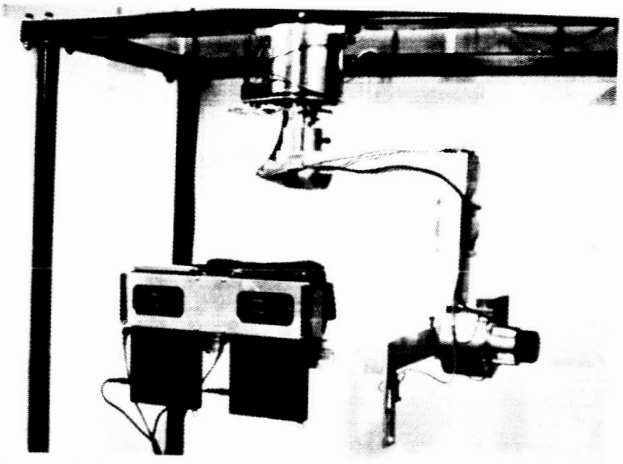


Fig. 4(a) Impedance controlled active display.

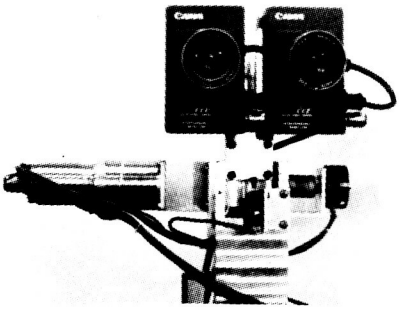


Fig. 4(b) Slave camera system.

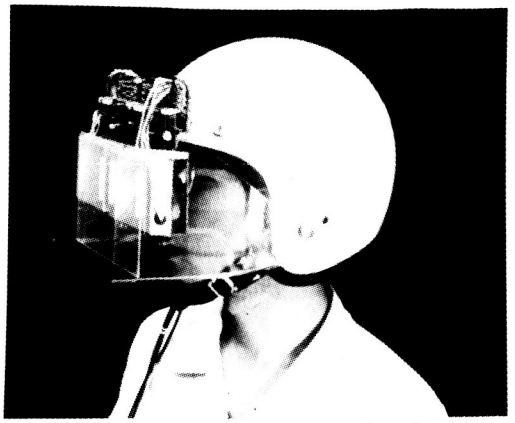


Fig. 5 Head mounted display.

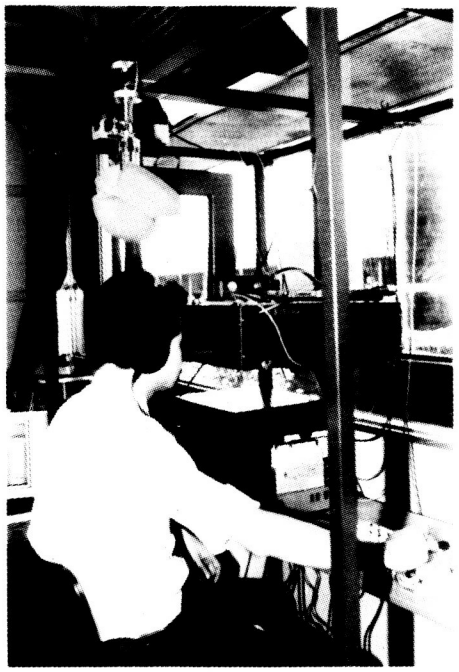


Fig. 6(a) Head-coupled display.

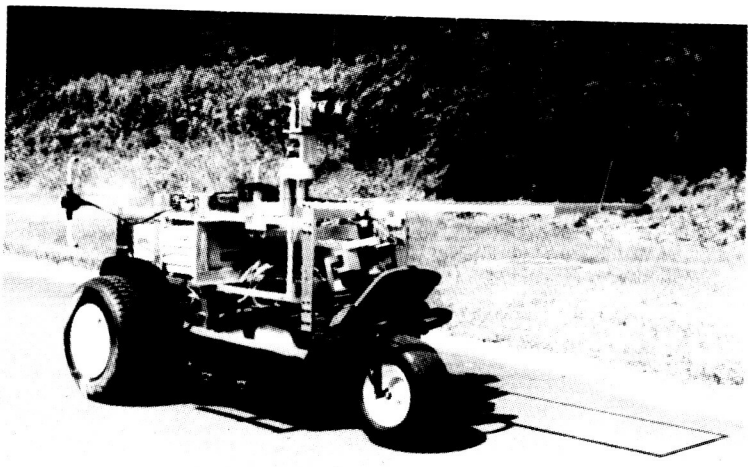


Fig. 6(b) Tele-existence mobile robot.

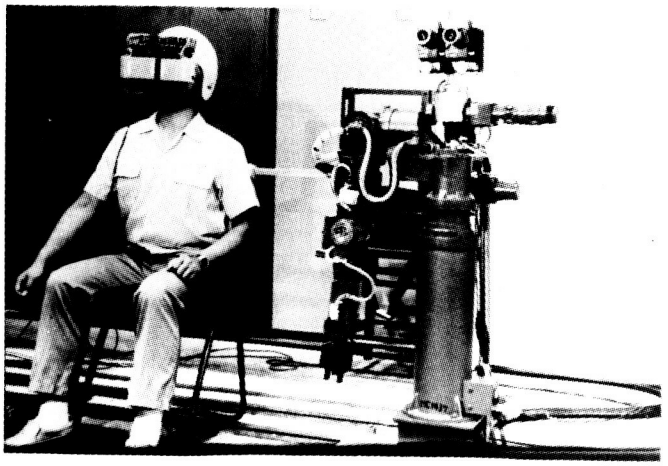


Fig. 7 Anthropomorphic slave robot.

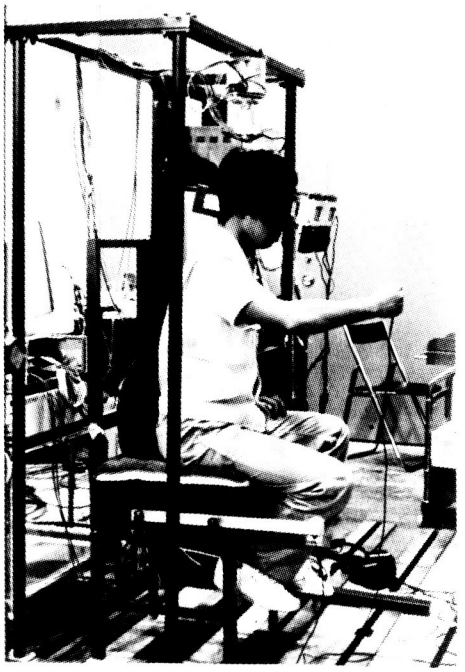


Fig. 8 Operation using impedance controlled display.

REDUNDANCY OF SPACE MANIPULATOR ON FREE-FLYING VEHICLE AND ITS NONHOLONOMIC PATH PLANNING

Yoshihiko Nakamura and Ranjan Mukherjee

Mechanical and Environmental Engineering and
Center for Robotic Systems in Microelectronics
University of California, Santa Barbara
California, 93106

ABSTRACT

This paper discusses the nonholonomic mechanical structure of space robots and its path planning. The angular momentum conservation works as a nonholonomic constraint while the linear momentum conservation is a holonomic one. Taking this in to account, a vehicle with a 6 d.o.f. manipulator is described as a 9 variable system with 6 inputs. This fact implies the possibility to control the vehicle orientation as well as the joint variables of the manipulator by actuating the joint variables only if the trajectory is carefully planned, although both of them cannot be controlled independently. It means that assuming feasible-path planning a system that consists of a vehicle and a 6 d.o.f. manipulator can be utilized as 9 d.o.f system. In this paper, first, the nonholonomic mechanical structure of space vehicle/manipulator system is shown. Second, a path planning scheme for nonholonomic systems is proposed using Lyapunov functions.

1. INTRODUCTION

The control of space vehicle/manipulator system possesses inherent issues that have not been considered for on-the-earth robot manipulators, such as the micro gravity, momentum conservation, and preciousness of energy. The kinematics and dynamics of space vehicle/manipulator systems have recently been studied by various researchers.

Alexander and Cannon [1] assumed concurrent use of the thrust force of vehicle and the manipulator joint torque, and proposed a control scheme taking account of the effect of the thrust force in computing the joint torque of manipulator. Dobowsky and Vafa [2] and Vafa [3] proposed a novel concept to simplify the kinematics and dynamics of space vehicle/manipulator system. A *virtual manipulator* is an imaginary manipulator that has similar kinematic and dynamic structure to the real vehicle/manipulator system but fixed at the total center of mass of the system. By solving the motion of the virtual manipulator for the desired motion of endeffector, the motion of vehicle/manipulator system is obtained straightforwardly. On the other hand, Umetani and Yoshida [4] reported a method to continuously control the motion of endeffector without actively controlling the vehicle. The momentum conservations for linear and angular motion are explicitly represented and used as the constraint equations to eliminate dependent variables and obtain the generalized Jacobian matrix that relates the joint motion and the endeffector motion. Longman, Lindberg, and Zadd [5] also discussed the coupling of manipulator motion and vehicle motion. Miyazaki, Masutani, and Arimoto [6] discussed a sensor feedback scheme using the transposed generalized Jacobian matrix.

Both of the linear and angular momentum conservations have been used to eliminate dependent variables[4] [6]. Although both of them are represented by equations of velocities, the linear one can be exhibited by the motion of the center of mass of the total system, which is represented by the equations of positions not of velocities. This implies that the linear momentum conservation is integrable and hence a holonomic constraint. On the other hand, the angular momentum conservation cannot be represented by an integrated form, which means that it is a nonholonomic constraint.

Suppose an n d.o.f. manipulator on a vehicle, the motion of the endeffector is described by $n+6$ variables, n

of the manipulator and 6 of the vehicle. By eliminating holonomic constraint of linear momentum conservation, the total system is formulated as a nonholonomic system of n+3 variables including 3 dependent variables. Although only n variables out of n+3 can be independently controlled, with an appropriate path planning scheme it would possible to converge all of n+3 variables to a desired values due to the nonholonomic mechanical structure. A similar situation is experienced in our daily life. Although an automobile has two independent variables to control, that is, wheel rotation and steering, it can be parked at an arbitrary place with an arbitrary orientation in two dimensional space. This can be done because it is a nonholonomic system.

To locate the manipulator endeffector at a desired point with a desired orientation, even a vehicle with a 6 d.o.f. manipulator has redundancy because a variety of vehicle orientation can be chosen at the final time. This kind of nonholonomic redundancy would be utilized (1) when the extended Jacobian control results in an infeasible motion due to the physical joint limitation, (2) when the system requires more degrees of freedom to avoid obstacles at the final location of the endeffector, (3) when the vehicle orientation needs to be changed without using propulsion or a momentum gyro, and so on.

In this paper, we propose a path planning scheme to control both of the vehicle orientation and the manipulator joints by actuating manipulator joints only. First, the nonholonomic mechanical structure of space vehicle/manipulator system is shown. Second, a path planning scheme for nonholonomic systems is proposed using Lyapunov functions. Since the planning scheme is given in a general form, it can be applied to other many nonholonomic planning problems, such as the path planning of 2 d.o.f. vehicles for 3 d.o.f. motion in a plane, planning of contact point motion of multifingered hands with spherical rolling contacts, and so on.

2. ANGULAR MOMENTUM CONSERVATION AS NONHOLONOMIC CONSTRAINT

2.1 Nomenclature

frame I	Inertia frame.
frame V	Vehicle frame.
frame B	Manipulator base frame
frame E	Manipulator endeffector frame
frame K	k-th body frame. k-th link frame of manipulator for $k = 1, \cdots, n$. n-th link frame is identical to the manipulator endeffector frame. Vehicle frame for $k = 0$.
m_k	Mass of the k-th body (kg). 0-th body is the vehicle. k-th body ($k \geq 1$) is the k-th link of the manipulator.
${}^I\boldsymbol{r}_k \in R^3$	Position vector from the origin of the inertia frame to the center of mass of k-th body represented in theinertia frame. (m)
${}^B\boldsymbol{r}_k \in R^3$	Position vector from the origin of the manipulator base frame to the center of mass of k-th body represented in the manipulator base frame.(m)
${}^I\boldsymbol{\omega}_k \in R^3$	Angular velocity of the k-th body in the inertia frame.(rad/s)
${}^k\boldsymbol{I}_k \in R^{3\times3}$	Inertia matrix of the k-th body about its center of mass in the k-th body frame. (kgm^2)
${}^I\boldsymbol{I}_k \in R^{3\times3}$	Inertia matrix of the k-th body about its center of mass in the inertia frame. (kgm^2)
$\dot{\boldsymbol{\theta}}_1 \in R^6$	Linear velocity of the center of mass and angular velocity of the vehicle in inertia frame. (m/s, rad/s)
$\boldsymbol{\theta}_2 \in R^n$	Joint variables $(q_1, \cdots, q_n)$ of the manipulator. (rad)
${}^I\boldsymbol{A}_B \in R^{3\times3}$	Rotation matrix from the inertia frame to the manipulator base frame.
${}^I\boldsymbol{A}_k \in R^{3\times3}$	Rotation matrix from the inertia frame to thek-th body frame (vehicle frame for $k = 0$, k-th link frame of the manipulator for $k = 1, \cdots, n$).
$\boldsymbol{J}_2^k \in R^{3\times n}$	Jacobian matrix of the position of the center of mass of k-th body ($k = 1, \cdots, n$) in the manipulator base frame. (m)
$\boldsymbol{E}_i \in R^{i\times i}$	$i \times i$ identity matrix.
α, β, γ	z-y-x Euler angles.

2.2 Kinematics of Space Vehicle/Manipulator System

The basic equations of kinematics of space vehicle/manipulator system is developed in this subsection. Fig. 1 shows a model of space vehicle/manipulator system. Five kinds of frames, the inertia frame, the vehicle frame, the manipulator base frame, the k-th link frames, and the manipulator endeffector frame, are represented by I, V, B, K, and E respectively. The link frames of the manipulator are defined by Denavit-Hartenberg convention [7]. The vehicle frame is assumed to be fixed at the center of mass of the vehicle.

Supposing zero linear and angular momentum at initial time, the linear and angular momentum conservations are represented by

$$\sum_{k=0}^{n} m_k {}^I\dot{\boldsymbol{r}}_k = 0, \tag{1}$$

$$\sum_{k=0}^{n} ({}^I\boldsymbol{I}_k {}^I\boldsymbol{\omega}_k + m_k {}^I\boldsymbol{r}_k \times {}^I\dot{\boldsymbol{r}}_k) = 0, \tag{2}$$

The vehicle and manipulator motions are described by the following $\dot{\boldsymbol{\theta}}_1$ and $\dot{\boldsymbol{\theta}}_2$.

$$\dot{\boldsymbol{\theta}}_1 = \begin{pmatrix} {}^I\dot{\boldsymbol{r}}_0 \\ {}^I\boldsymbol{\omega}_0 \end{pmatrix} \tag{3}$$

$$\boldsymbol{\theta}_2 = \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} \tag{4}$$

${}^I\dot{\boldsymbol{r}}_k$ is computed by

$$\begin{aligned} {}^I\dot{\boldsymbol{r}}_k &= {}^I\dot{\boldsymbol{r}}_0 + {}^I\boldsymbol{\omega}_0 \times ({}^I\boldsymbol{r}_k - {}^I\boldsymbol{r}_0) + {}^I\boldsymbol{A}_B \boldsymbol{J}_2^k \dot{\boldsymbol{\theta}}_2 \\ &= (\boldsymbol{E}_3 \quad -{}^I\boldsymbol{R}_{0k})\,\dot{\boldsymbol{\theta}}_1 + {}^I\boldsymbol{A}_B\, \boldsymbol{J}_2^k\, \dot{\boldsymbol{\theta}}_2 \end{aligned} \tag{5}$$

where ${}^I\boldsymbol{R}_{0k}$ and ${}^I\boldsymbol{r}_{0k}$ are defined by

$${}^I\boldsymbol{R}_{0k} = \begin{pmatrix} 0 & -{}^Ir_{0kz} & {}^Ir_{0ky} \\ {}^Ir_{0kz} & 0 & -{}^Ir_{0kx} \\ -{}^Ir_{0ky} & {}^Ir_{0kx} & 0 \end{pmatrix} \tag{6}$$

$${}^I\boldsymbol{r}_k - {}^I\boldsymbol{r}_0 = \begin{pmatrix} {}^Ir_{0kx} \\ {}^Ir_{0ky} \\ {}^Ir_{0kz} \end{pmatrix} \tag{7}$$

On the other hand, ${}^I\boldsymbol{I}_k {}^I\boldsymbol{\omega}_k$ is given by

$${}^I\boldsymbol{I}_k {}^I\boldsymbol{\omega}_k = {}^I\boldsymbol{A}_k {}^k\boldsymbol{I}_k {}^I\boldsymbol{A}_k{}^T {}^I\boldsymbol{\omega}_k \tag{8}$$

$${}^I\boldsymbol{\omega}_k = \begin{cases} (0 \quad \boldsymbol{E})\,\dot{\boldsymbol{\theta}}_1 & \text{for } k = 0 \\ {}^I\boldsymbol{\omega}_0 + \sum_{j=1}^{k} {}^I\boldsymbol{A}_j \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{q}_j & \text{for } k = 1, \cdots, n \end{cases} \tag{9}$$

By substituting eqs. (5) and (8) into eqs. (1) and (2) and summarizing them in a matrix form, the linear and angular momentum conservations are represented by the following equation.

$$\boldsymbol{H}_1 \dot{\boldsymbol{\theta}}_1 + \boldsymbol{H}_2 \dot{\boldsymbol{\theta}}_2 = 0 \tag{10}$$

$$\boldsymbol{H}_1 = \begin{pmatrix} \sum_{k=0}^{n} m_k \boldsymbol{E} & -\sum_{k=0}^{n} m_k {}^I\boldsymbol{R}_{0k} \\ \sum_{k=0}^{n} m_k {}^I\boldsymbol{R}_k & \sum_{k=0}^{n} {}^I\boldsymbol{A}_k {}^k\boldsymbol{I}_k {}^I\boldsymbol{A}_k{}^T - \sum_{k=0}^{n} m_k {}^I\boldsymbol{R}_k {}^I\boldsymbol{R}_{0k} \end{pmatrix} \tag{11}$$

$$\boldsymbol{H}_2 = \begin{pmatrix} \sum_{k=0}^{n} m_k {}^I\boldsymbol{A}_B \boldsymbol{J}_2^k \\ \sum_{k=0}^{n} m_k {}^I\boldsymbol{R}_k {}^I\boldsymbol{A}_B \boldsymbol{J}_2^k + \boldsymbol{P} \end{pmatrix} \tag{12}$$

where

$${}^I\boldsymbol{R}_k = \begin{pmatrix} 0 & -{}^Ir_{kz} & {}^Ir_{ky} \\ {}^Ir_{kz} & 0 & -{}^Ir_{kx} \\ -{}^Ir_{ky} & {}^Ir_{kx} & 0 \end{pmatrix} \tag{13}$$

$$\boldsymbol{P} = (\boldsymbol{P}_1 \quad \boldsymbol{P}_2 \quad \cdots \quad \boldsymbol{P}_n) \tag{14}$$

$$\boldsymbol{P}_i = \left(\sum_{k=i}^{n} {}^I\boldsymbol{A}_i {}^i\boldsymbol{I}_i {}^I\boldsymbol{A}_i{}^T \right) {}^I\boldsymbol{A}_i \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

In eq. (13), ${}^Ir_{kx}$, ${}^Ir_{ky}$ and ${}^Ir_{kz}$ are x, y and z components of ${}^I\boldsymbol{r}_k$ respectively.

The relationship between the endeffector, $\dot{\boldsymbol{\theta}}_1$ and $\dot{\boldsymbol{\theta}}_2$ is described in the following form.

$$\dot{\boldsymbol{h}} = \boldsymbol{J}_1\dot{\boldsymbol{\theta}}_1 + \boldsymbol{J}_2\dot{\boldsymbol{\theta}}_2 \tag{15}$$

where

$$\dot{\boldsymbol{h}} = \begin{pmatrix} {}^I\dot{\boldsymbol{r}}_E \\ {}^I\boldsymbol{\omega}_E \end{pmatrix}$$

$\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are the pure geometrical Jacobian matrices which do not take account of the momentum conservations. In eq. (10), $\boldsymbol{H}_1 \in R^{6\times 6}$ is always nonsingular. Therefore, eq. (10) is identical to

$$\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{H}_1{}^{-1}\boldsymbol{H}_2\dot{\boldsymbol{\theta}}_2 \tag{16}$$

Substituting eq. (16) into eq. (15) offers

$$\dot{\boldsymbol{h}} = \left(-\boldsymbol{J}_1\boldsymbol{H}_1{}^{-1}\boldsymbol{H}_2 + \boldsymbol{J}_2\right)\dot{\boldsymbol{\theta}}_2 \tag{17}$$

Umetani and Yoshida [4] named the coefficient matrix of the above equation *the generalized Jacobian matrix*. In this derivation, the momentum conservations of eq. (10) are used as constraints equations and eliminated in the final equation.

2.3 Holonomic and Nonholonomic Constraints

Eq. (1) can be analytically integrated as follows:

$$\begin{aligned} \int_0^t \sum_{k=0}^{n} m_k {}^I\dot{\boldsymbol{r}}_k \, dt &= \sum_{k=0}^{n} m_k {}^I\boldsymbol{r}_k(t) - \sum_{k=0}^{n} m_k {}^I\boldsymbol{r}_k(0) \\ &= 0 \end{aligned} \tag{18}$$

The above equation physically means that the total center of mass of the system does not move. ${}^I\boldsymbol{r}_k$ is computed by

$$^{I}\boldsymbol{r}_{k} = {}^{I}\boldsymbol{A}_{B}\,{}^{B}\boldsymbol{r}_{k} + {}^{I}\boldsymbol{r}_{0} \tag{19}$$

where ${}^{I}\boldsymbol{A}_{B}$ is a function of the vehicle orientation only. ${}^{B}\boldsymbol{r}_{k}$ is a function of the joint variables of the manipulator only. Knowing the vehicle orientation, the joint variables, and the initial position of the total center of mass, the vehicle position ${}^{I}\boldsymbol{r}_{k}$ can be obtained by substituting eq. (19) into eq. (18). Therefore, the linear momentum conservation is considered a holonomic constraint because it is integrable.

Although eqs. (1) and (2) are both represented by velocities, eq. (2) can not be analytically integrated and, therefore, it is a nonholonomic constraint. The physical characteristic of nonholonomic constraint is exhibited by the fact that even if the manipulator joints return to the initial joint variables after a sequence of motion, the vehicle orientation may not be the same as its initial value. The vehicle orientation can be eliminated as a dependent variable as we did in deriving eq. (17). In next section, we propose to control both of the independent and dependent variables by controlling the independent ones only.

The basic system equation is obtained by taking the vehicle orientation and $\boldsymbol{\theta}_2$ as the state variable and the $\dot{\boldsymbol{\theta}}_2$ as the input variable. First, the coefficient matrix of eq. (16) is divided into a top $3 \times n$ matrix and a bottom $3 \times n$ matrix as follows:

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_r \\ \boldsymbol{H}_\omega \end{pmatrix} = -\boldsymbol{H}_1^{-1}\boldsymbol{H}_2 \tag{20}$$

The state variable $\boldsymbol{x}$ and the input variable $\boldsymbol{u}$ are defined by

$$\boldsymbol{x} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \\ \boldsymbol{\theta}_2 \end{pmatrix} \in R^{n+3} \tag{21}$$

$$\boldsymbol{u} = \dot{\boldsymbol{\theta}}_2 \in R^{n} \tag{22}$$

α,β, and γ are the z-y-x Euler angles of the vehicle with respect to the inertia frame. The relationship between the Euler angles and ${}^{I}\boldsymbol{\omega}_0$ is given by

$$^{I}\boldsymbol{\omega}_0 = \boldsymbol{N}\begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} \tag{23}$$

where

$$\boldsymbol{N} = \begin{pmatrix} 0 & -sin\alpha & cos\alpha\, cos\beta \\ 0 & cos\alpha & sin\alpha\, \cos\beta \\ 1 & 0 & -sin\beta \end{pmatrix}$$

The system equation becomes

$$\dot{\boldsymbol{x}} = \boldsymbol{K}\,\boldsymbol{u} \tag{24}$$

where

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{N}^{-1}\boldsymbol{H}_\omega \\ \boldsymbol{E}_n \end{pmatrix} \in R^{(n+3)\times n} \tag{25}$$

2.4 Nonholonomic Redundancy

The system represented by eq. (25) has a unique feature in the fact that the input variable may not be found even if a smooth desired trajectory of $\boldsymbol{x}$ is provided because it has less number of input variable. It is impossible to plan a feasible trajectory without taking full account of the dynamics of eq. (25). This is a general feature of nonholonomic mechanical systems. An automobile can move around in two dimensional space and

orient itself if we drive it properly, although it has only two variables to control, that is, wheel rotation and steering. In this case, the state variables are three and the inputs are two.

By appropriately planning the trajectory, the desired final values of the vehicle orientation and the manipulator joint variables could be reached. To locate the manipulator endeffector at a desired point with a desired orientation, even a vehicle with a 6 d.o.f. manipulator has redundancy because a variety of vehicle orientation can be chosen at the final time. The choice of the final vehicle orientation can be done based on the conventional control or planning schemes of kinematically redundant manipulators [8, 9, 10]. It is a problem to find an appropriate configuration among the configurations attained by 3 d.o.f. selfmotion.

The nonholonomic redundancy would be utilized (1) when the extended Jacobian control results in an infeasible motion due to the physical joint limitation, (2) when the system requires more degrees of freedom to avoid obstacles at the final location of the endeffector, (3) when the vehicle orientation needs to be changed without using propulsion or a momentum gyro, and so on.

3. PATH PLANNING USING LYAPUNOV FUNCTIONS

3.1 First Lyapunov function

In this section, the input variable $\boldsymbol{u}$ is synthesized based on the Lyapunov's direct method [11] so that the vehicle orientation and the joint variables should converge to their desired values.

The following function is chosen as a candidate of the Lyapunov function.

$$v_1 = \frac{1}{2}\Delta\boldsymbol{x}^T\boldsymbol{A}\Delta\boldsymbol{x} \tag{26}$$

$$\Delta\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{x} \tag{27}$$

where $\boldsymbol{A}$ is a positive definite constant matrix. $v_1 = 0$ is attained only when $\boldsymbol{x}_d = \boldsymbol{x}$. The time derivative of v_1 is computed as follows:

$$\dot{v}_1 = -\Delta\boldsymbol{x}^T\boldsymbol{A}\dot{\boldsymbol{x}} = -\Delta\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{K}\boldsymbol{u} \tag{28}$$

where eq. (24) was substituted. Now, choosing the input variable as

$$\boldsymbol{u}_1 = (\boldsymbol{A}\boldsymbol{K})^T\Delta\boldsymbol{x}, \tag{29}$$

the rate of change of the Lyapunov function becomes

$$\dot{v}_1 = -\boldsymbol{u}_1^T\boldsymbol{u}_1 \leq 0 \tag{30}$$

If the equality of eq. (30) holds only when $\boldsymbol{x}_d = \boldsymbol{x}$, Lyapunov's theorem [11] can conclude its global stability. However, eq. (30) is not the case. $\dot{v}_1$ becomes zero when $\Delta\boldsymbol{x}$ is in the null space of $(\boldsymbol{A}\boldsymbol{K})^T$, which is a three dimensional space.

3.2 Avoiding Null Space of $(\boldsymbol{A}\boldsymbol{K})^T$

The LaSalle's theorem [12] says that the state variable $\boldsymbol{x}$ converges to $\boldsymbol{x}_d$ if $\boldsymbol{x} = \boldsymbol{x}_d$ is the unique entry of the maximum invariant set. When $\Delta\boldsymbol{x}$ is at the null space of $(\boldsymbol{A}\boldsymbol{K})^T$ and it stays within the null space thereafter, all the points on this trajectory are the entries of the maximum invariant set. In this subsection, the unit vector is chosen such that $\Delta\boldsymbol{x}$ should avoid the null space as much as possible and get out of the null space if it is there.

To take account of the null space of $(\boldsymbol{A}\boldsymbol{K})^T$ we introduce the second Lyapunov function v_2 such that

$$v_2 = \frac{\Delta\boldsymbol{x}^T\left(\boldsymbol{I} - (\boldsymbol{A}\boldsymbol{K})(\boldsymbol{A}\boldsymbol{K})^{\#}\right)\Delta\boldsymbol{x}}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x} + \epsilon_1} \tag{31}$$

where ϵ_1 is a positive small constant. v_2 becomes equal to zero when $\Delta\boldsymbol{x} = 0$.

Since

$$\begin{aligned}&\Delta \boldsymbol{x}^T \left(\boldsymbol{I} - (\boldsymbol{AK})(\boldsymbol{AK})^{\#}\right) \Delta \boldsymbol{x} \\ =&\Delta \boldsymbol{x}^T \left(\boldsymbol{I} - (\boldsymbol{AK})(\boldsymbol{AK})^{\#}\right)^T \left(\boldsymbol{I} - (\boldsymbol{AK})(\boldsymbol{AK})^{\#}\right) \Delta \boldsymbol{x} \\ =&\| \left(\boldsymbol{I} - (\boldsymbol{AK})(\boldsymbol{AK})^{\#}\right) \Delta \boldsymbol{x} \|^2,\end{aligned} \tag{32}$$

the numerator of eq. (31) implies the squared Euclidean norm of the orthogonal projection of $\Delta \boldsymbol{x}$ on the null space of $(\boldsymbol{AK})^T$. If we define ϕ such that

$$cos\,\phi = \frac{\| \left(\boldsymbol{I} - (\boldsymbol{AK})(\boldsymbol{AK})^{\#}\right) \Delta \boldsymbol{x} \|}{\| \Delta \boldsymbol{x} \|}, \qquad 0 \leq \phi \leq \frac{\pi}{2} \tag{33}$$

ϕ means an angle between $\Delta \boldsymbol{x}$ and the hyperplane of the null space of $(\boldsymbol{AK})^T$, and can be considered as a distance of $\Delta \boldsymbol{x}$ from the null space as shown in Fig. 2. For $\epsilon_1 = 0$ the second Lyapunov function becomes

$$v_2 = cos^2\,\phi \qquad 0 \leq \phi \leq \frac{\pi}{2}. \tag{34}$$

In eq. (31), ϵ_1 allows for v_2 not to take extreme values and to be defined at $\Delta \boldsymbol{x}$. In eq. (34) v_2 is monotonously reduced as ϕ grows, and takes zero at $\phi = \pi/2$, which means the farthest point from the null space.

Taking the derivative of v_2 with respect to time, we have

$$\dot{v}_2 = \frac{\partial v_2}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = \frac{\partial v_2}{\partial \boldsymbol{x}} \boldsymbol{K} \boldsymbol{u}. \tag{35}$$

If we choose $\boldsymbol{u}_2$ such as

$$\boldsymbol{u}_2 = -\boldsymbol{K}^T \left(\frac{\partial v_2}{\partial \boldsymbol{x}}\right)^T, \tag{36}$$

and use it as $\boldsymbol{u}$, then $\dot{v}_2 \leq 0$, and $\boldsymbol{u}_2$ works to avoid the null space by driving toward $\phi = \pi/2$.

We integrate $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ in a hierarchical manner such that

$$\boldsymbol{u} = k_1 \boldsymbol{u}_1 + k_2 \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right) \boldsymbol{u}_2 \tag{37}$$

where $\boldsymbol{u}_1^{\#}$ is the pseudoinverse of $\boldsymbol{u}_1$, k_1 and k_2 are positive constants. Since $\left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right) \boldsymbol{u}_2$ is the orthogonal projection of $\boldsymbol{u}_2$ onto the hyperplane that is perpendicular to $\boldsymbol{u}_1$, the first and second terms are mutually perpendicular. This results in following properties of eq. (37).

The second term of eq.(37) has no effect on the convergence speed of v_1 † because substituting eq. (37) into eq. (28) we have

$$\dot{v}_1 = -\boldsymbol{u}_1^T \{\boldsymbol{u}_1 + k_1 \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}^{\#}\right) \boldsymbol{u}_2\} = -\boldsymbol{u}_1^T \boldsymbol{u}_1 \tag{38}$$

where $\boldsymbol{u}_1^T \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right) = \left(\boldsymbol{u}_1 - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#} \boldsymbol{u}_1\right)^T = 0$ is used.

Let's consider the effect of the second term of v_2. Substituting the second term of eq. (37) into eq. (35) along with eq. (36), we obtain

$$\begin{aligned}\dot{v}_2 &= -\frac{\partial v_2}{\partial \boldsymbol{x}} \boldsymbol{K} \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right) \boldsymbol{K}^T \left(\frac{\partial v_2}{\partial \boldsymbol{x}}\right)^T \\ &= -\frac{\partial v_2}{\partial \boldsymbol{x}} \boldsymbol{K} \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right)^T \left(\boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^{\#}\right) \boldsymbol{K}^T \left(\frac{\partial v_2}{\partial \boldsymbol{x}}\right)^T \leq 0\end{aligned} \tag{39}$$

† The convergence speed $\dot{v}_1$ is the same for both $\boldsymbol{u}_1$ and $\boldsymbol{u}$ of eq. (37) only in local sense. Since the global trajectory of $\boldsymbol{x}$ varies depending on the choice of the input, the global convergence speed would be different.

$\dot{v}_2$ becomes zero only when $(\boldsymbol{I} - \boldsymbol{u}_1\boldsymbol{u}_1^{\#})\,\boldsymbol{K}^T\,(\partial v_2/\partial \boldsymbol{x})^T = 0$. Otherwise $\dot{v}_2$ is always negative. This means that the second term of eq. (37) tries to reduce v_2 although the total $\boldsymbol{u}$ of eq. (37) does not be guarantee the negativeness of $\dot{v}_2$ because of the effect of the first term.

To summarize eqs. (28),(29),(35),and (36), the proposed hierarchical Lyapunov function approach can be represented as follows

$$\boldsymbol{u} = k_1\,\boldsymbol{u}_1 + k_2\,(\boldsymbol{I} - \boldsymbol{u}_1\boldsymbol{u}_1^{\#})\,\boldsymbol{u}_2 \tag{40}$$

$$\boldsymbol{u}_i = -\boldsymbol{K}^T\left(\frac{\partial v_i}{\partial \boldsymbol{x}}\right)^T, \qquad \text{for } i = 1,2 \tag{41}$$

It should be noted that if we consider v_i as the ith manipulation variable, $(\partial v_i/\partial \boldsymbol{x})\,\boldsymbol{K}$ as its Jacobian matrix with respect to the input variable $\boldsymbol{u}$, then eq. (40) is identical to the *task-priority approach* developed for kinematically redundant manipulators[10], having

$$\dot{v}_i = -\frac{\partial v_i}{\partial \boldsymbol{x}}\boldsymbol{K}\boldsymbol{K}^T\left(\frac{\partial v_i}{\partial \boldsymbol{x}}\right)^T, \qquad \text{for } i = 1,2 \tag{42}$$

as the desired trajectories of the manipulation variables. This approach cannot guarantee that $\boldsymbol{x} = \boldsymbol{x}_d$ is the unique entry of the maximum invariant set [12] and, therefore, the trajectory may halt at some point in the null space of $(\boldsymbol{A}\boldsymbol{K})^T$. However, if the second Lyapunov function can successfully avoid the null space of $(\boldsymbol{A}\boldsymbol{K})^T$, $\boldsymbol{x}$ converges to $\boldsymbol{x}_d$.

4. CONCLUSION

A new insight of the mechanical structure of space vehicle/ manipulator systems was given. By utilizing the nonholonomic structure, not only the manipulator joints, but also vehicle orientation can be controlled only by actuating the joint variables, although both of the vehicle motion and the manipulator joints cannot be controlled independently. Therefore, it is essential to plan a feasible trajectory. A nonlinear control scheme was synthesized using Lyapunov's direct method. This scheme can be used not only for real-time control, but for planning of a feasible motions of vehicle and manipulator. To verify the effectiveness of the proposed approach, numerical simulation is currently being undertaken at the Center for Robotic Systems in Microelectronics, University of California, Santa Barbara.

ACKNOWLEDGEMENT

This material is supported by the National Science Foundation under Contract number 8421415. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the Foundation.

REFERENCES

[1] Alexander, H.L., and Cannon, R.H.,"Experiments on the Control of a Satellite Manipulator." Proceedings of 1987 American Control Conference, Seattle, WA, June 1987.

[2] Dobowsky, S., and Vafa, Z., "A Virtual Manipulator Model for Space Robotic Systems." Proceedings of Workshop on Space Telerobotics, JPL Publication 87-13, Vol. 3, pp. 335-342, Pasadena, CA, January, 1987.

[3] Vafa, Z., "The Kinematics, Dynamics and Control of Space Manipulators: The Virtual Manipulator Concept." Ph.D. Dissertation in Mechanical Engineering, MIT,1987.

[4] Umetani, Y., and Yoshida, K., "Continuous Path Control of Space Manipulators Mounted on OMV." Acta Astronautica, Vol. 15, No.12, pp. 981-986, 1987.

[5] Longman, R.W., Lindberg, R.E., and Zadd, M.F., "Satellite-Mount-ed Robot Manipulators- New Kinematics and Reaction Moment Compensation." International Journal of Robotics Research, Vol.6, No.3, pp. 87-103, 1987.

[6] Miyazaki, F., Masutani, Y., and Arimoto, S., "Sensor Feedback Using Approximate Jacobian." Proceedings of USA-Japan Symposium on Flexible Automation, pp. 139-145, Minneapolis, July, 1988.
[7] Denavit, J., and Hartenberg, R.S., "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices." Journal of Applied Mechanics, Vol.22, 1955.
[8] Ligeois, A., "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms." IEEE Trans. System, Man and Cybernetics, Vol. SMC-7, No. 12, pp. 868-871, 1977.
[9] Nakamura, Y., and Hanafusa, H., "Optimal Redundancy Control of Robot Manipulators." International Journal of Robotics Research, Vol.6, No.1, pp. 32-42, 1987.
[10] Nakamura, Y., Hanafusa, H., and Yoshikawa, T., "Task-Priority Based Redundancy Control of Robot Manipulators." International Journal of Robotics Research, Vol.6, No.2, pp. 3-15, 1987.
[11] Lyapunov, A.M., "On the General Problem of Stability of Motion." Soviet Union: Kharkov Mathematical Society, 1892 *(in Russian)*.
[12] LaSalle, J., and Lefschetz, S., ***Stability by Lyapunov's Direct Method with Applications.*** New York: Academic Press, 1961.

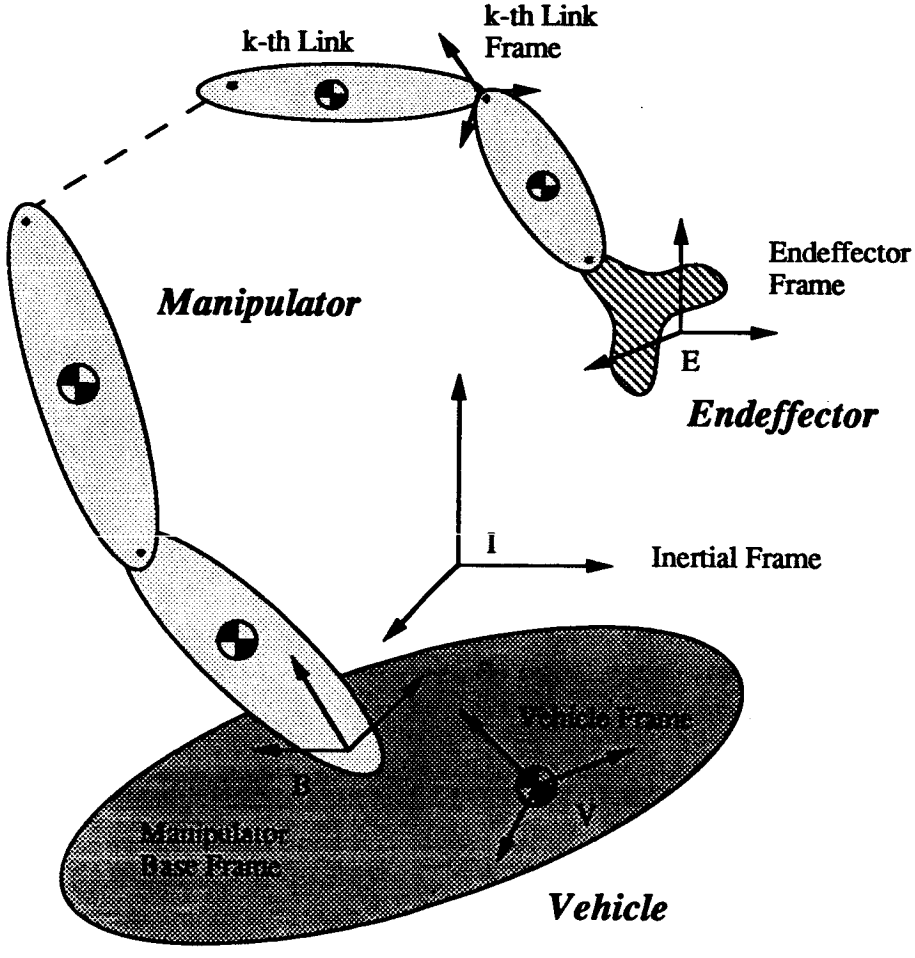
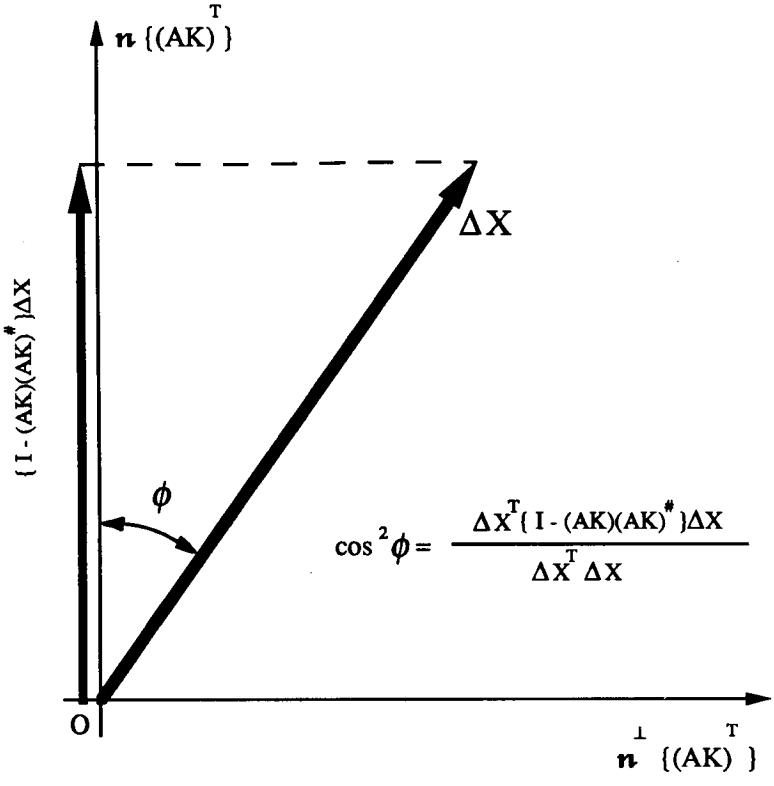


Fig 1. Five Coordinate Frames for the Space Vehicle / Manipulator System.



$n\{(AK)^T\}$: the null space of $(AK)^T$

$n^{\perp}\{(AK)^T\}$: the orthogonal complement space of $n\{(AK)^T\}$

Fig 2. Physical Meaning of the 2nd Lyapunov Function and Definition of Angle ϕ .

Guidance Algorithms for a Free-Flying Space Robot

A.F. Brindle, H.E.M. Viggh, J.H. Albert
Boeing Aerospace
P.O. Box 3999 MS 82-58
Seattle, WA 98124

Abstract

Robotics is a promising technology for assembly, servicing, and maintenance of platforms in space. This project is investigating several aspects of planning and guidance for telesupervised and fully autonomous robotic servicers. This paper describes ongoing work on guidance algorithms for proximity operations of a free flyer. The general approach combines numeric trajectory optimisation with artificial intelligence based obstacle avoidance. An initial algorithm and results of simulating it on a platform servicing scenario are discussed. A second algorithm experiment is then proposed.

Keywords: Autonomous robotics, spacecraft servicing, artificial intelligence, trajectory optimisation, obstacle avoidance.

1 Introduction

Robotics is a promising technology for assembly, servicing, and maintenance of platforms in space. Robots will reduce expensive and risky astronaut extra-vehicular activity (EVA) around manned systems and permit servicing of remote platforms. Several robots, for example, have been proposed for the Space Station (see the reports of the NASA Advanced Technology Advisory Committee [1] and subsequent studies, including the Phase B Automation and Robotics Plan for Work Package 1 [2], which has an excellent bibliography.) Proposals for spacecraft servicers include teleoperated, telesupervised, and autonomous robots. Teleoperations will be the first step in deploying on-orbit robots, but long term goals include telesupervision and perhaps full autonomy for servicers.

This project is investigating several aspects of planning and guidance for a telesupervised and autonomous robotic servicers. This paper describes work on guidance algorithms for proximity operations of a free flyer. First, the selection of a servicing scenario is described and the functional focus of the project defined. Then the general project approach to developing guidance algorithms is presented. The strategy is to combine numeric trajectory optimization with artificial intelligence (AI) based obstacle avoidance to provide autonomous motion between specified start and goal points. Such guidance would be essential for autonomous operations and would form a component of a telesupervised system. The first algorithm and results of simulating it on the scenario are discussed. Finally, a proposal is given for a second algorithm experiment.

2 Robotic Spacecraft Servicer Scenario Definition

The domain of spacecraft servicing scenarios is large and varied. We considered these requirements for the project scenario:

- The scenario should address the needs of the space community roughly 20 years hence.
- The scenario should drive advances in AI and guidance technology.
- The scenario should not impose requirements on other entities in the environment to facilitate robot operation.

A note is needed about the last requirement. In most scenarios, it is possible to restrict or control other entities and thus ease requirements on the servicing robot. For example, it may be mandated that whenever a robot exits its orbiting home base, there will be no other entities active around the base, or no entities around the base at all. Any platform to be serviced may be required to have predefined corridors of safe approach for a servicer. Restrictions such as these will undoubtably be imposed on other entities in the future; however, if they are not assumed at this point, this research will aid in determining which restrictions are needed.

The next subsection defines a scenario in terms of the degree of autonomy assumed for the robot, the location of servicing, the type of target platform, and the complexity of the surrounding environment. The following subsection then describes the project focus within the selected scenario.

2.1 Scenario

A robotic servicer located in a bay of an orbiting home base is instructed to refuel and repair a remote platform. It is given a service order, such as a human maintenance engineer would receive, a description of the platform, including shape and current orbit (and probably where to obtain updates on the orbit), and a deadline for completing the work. It must obtain necessary supplies, exit the home base, perform orbit transfer, maneuver into proximity to the platform and inspect it from several points of view, dock, and service the platform.

The platform is a complex shape composed of trusses, solar panels, and antennae. Maneuvering around such a platform will require more sophisticated guidance than maneuvering around a small compact satellite. The home base is also a complex shape, and it is busy. Other robots and astronauts are moving around the base, and obstacles exist within its vicinity. The robot has to avoid multiple, moving, actively propelled objects.

2.2 Project Focus

This project focuses on guidance algorithms for the proximity operations performed by the robot near the home base and the platform. Proximity is defined as within 1000 feet. For these operations, the robot has a cold-gas thruster system similar to that of NASA's Manned Maneuvering Unit (MMU). To reflect this focus, the scenario is further defined to be an inspection task by a maintenance robot at the Space Station. Thus the home base and target platform are the same, and no orbit transfers are required.

There are four major functional components involved in proximity guidance.

1. Identification and tracking of platform and obstacles.
2. Generation of feasible goal positions and times.
3. Generation of robot trajectories.
4. Execution of trajectories, including reactive planning to handle contingencies.

These experiments are focused initially on the third function, trajectory generation. Therefore, a goal position and time have already been determined as input to the guidance module. The guidance system must plan a trajectory to this goal state which minimizes fuel and avoids foreseen obstacles.

It is assumed that this planning activity occurs prior to execution of the trajectory, while the robot is in a safe, stable position and able to sense the scene. Some path planning researchers argue that such *predictive* planning is of little utility for terrain vehicles and should be replaced by local, or *reactive*, responses to the immediate environment. This stems from the fact that robotic sensing and modeling of an uncertain and changing world is incomplete and inaccurate, leading to poor plans.

Certainly space robots will need reactive planning capability to respond to active obstacles, unforeseen obstacles, and other contingencies. However, there are two arguments in favor of having predictive planning as well. First, the environment is almost entirely synthetic, has a small number of understandable objects, and does not change rapidly. Accurate modeling of obstacles is quite conceivable. Second, there is a strong need for global optimization of trajectories. All space vehicles are fuel limited, so *a priori* identification of fuel efficient routes is highly desirable.

It is also assumed that object sensing and modeling can predict passive motions on orbit and can recognize regular motions of articulated parts on the target platform, with the help of the platform models supplied to it. Therefore, the scenario leads to the following requirements on trajectory planning by the guidance system:

1. The system generates a trajectory, or route plan, from a given start a given goal position.
2. The trajectory avoids all passive, foreseen obstacles, which may be in motion.
3. The trajectory avoids the platform of interest, which is a composite of different shapes.

3 Approach

The guidance algorithms must include 1) trajectory optimization to minimize fuel consumption, with orbital mechanics taken into account, and 2) obstacle avoidance. The general approach is to combine optimization techniques from the domain of spacecraft guidance with path planning and obstacle avoidance techniques from artificial intelligence (AI). Numerical optimization techniques cannot cope with the constraint explosion which occurs when more than one or two simple obstacles are modeled. AI has several techniques which handle multiple obstacles under various limiting assumptions, but these techniques have not been applied to on-orbit problems. By prototyping and

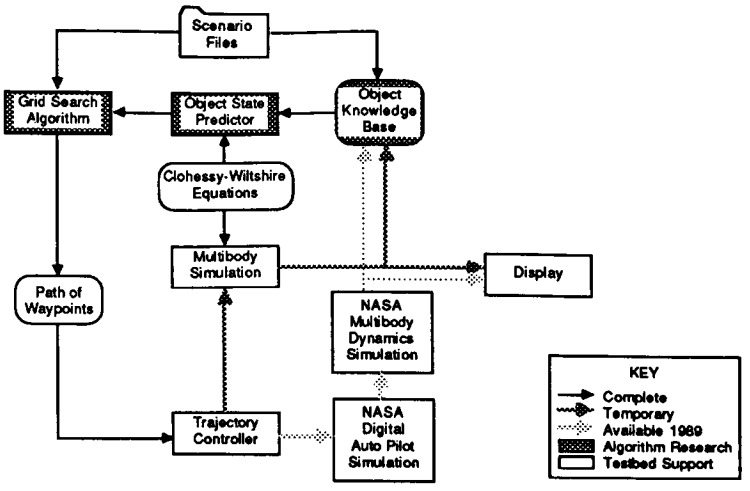


Figure 1: Software Simulation Testbed for Guidance Algorithm Experiments

evaluating algorithms which borrow techniques from both domains, we hope to arrive at a guidance system for safe, efficient on-orbit maneuvering.

The prototyping experiments are being performed on a Symbolics workstation. Several tools have been developed or obtained to form a simulation testbed for the work. The testbed is illustrated in Figure 1. The environment of platform and other objects is currently simulated on the Symbolics using Clohessy-Wiltshire equations for the orbital mechanics, but future experiments will use a multibody dynamics simulation acquired from NASA Johnson Space Center. This simulation has been installed on a VAX/UNIX system communicating with the Symbolics via Ethernet. The code includes a digital autopilot simulation for one of the bodies, which has been configured to match the MMU and will simulate the robot control and actuation systems for future closed loop experiments.

The simulations generate obstacle state information which is placed in the object knowledge base. Object shape information is loaded into the base from scenario files. The base controls concurrent access, so that simulation and planning can occur in separate processes.

The interface for algorithm demonstrations uses the Symbolics S-Geometry package for three dimensional, wire-frame graphical display. While displays have been generated on other, more graphics-oriented devices, hosting a 3-D display on the development machine was found to be indispensable for algorithm development and evaluation.

4 Grid Search Algorithm

The first experiment involved modifying a standard AI path planning technique to optimize fuel on-orbit, rather than the usual optimization of distance in two dimensions. Several AI techniques may extend to higher dimensionality, such as polyhedral blocks [7], or octrees [4,6] (see [3] for a survey). A* search [9] on a uniform grid, also called grid search [5,8], was selected for the initial implementation. This method is one of many which generate a space of points between the start and goal points. The space is searched for a set of waypoints which, when connected, form a safe, nearly optimal path from start to goal.

Table 1: Steps in the A* Search Algorithm

	Form a queue to hold partial paths from the start to other endpoints.
	Add the null path from start to start to the queue.
loop	If the queue is empty, return. failure to find path.
	Remove path P from the front of the queue.
	If P reaches the goal, return P as the solution.
	Form new paths by extending P toward the neighbors of P's endpoint.
	Add the new paths to the queue.
	Sort the queue by total cost, keeping lower cost paths in the front.
	If several paths in the queue have the same endpoint, discard all but the one with least cost.
	Repeat from loop.

A* search is based on discrete dynamic programming, which accomplishes breadth-first searching of a cost-weighted graph to find a shortest path. A* search differs from dynamic programming in that a heuristic factor is added to the cost function. A partial path's total cost is the sum of the cost of the path plus a heuristic estimate of the cost remaining from the endpoint of the path to the goal. The heuristic may render the solution less than optimal, but its use reduces search time. The algorithm is described in Table 1.

Adapting the grid search to space required new definitions of the cost function, the search space, the cost heuristic factor, and the grid with its notion of neighbor.

- The goal of minimizing fuel was approximated by using a cost function of magnitude of delta-v. The robot is assumed to fire its thrusters instantaneously, and only at the grid points. Moving from point A to point B is modeled as a single burn (thruster firing) at point A, and the magnitude of the change in velocity at point A is the cost of moving from A to B.

  There may be goals imposed on the robot in addition to minimizing fuel. To permit experiments which minimize time or distance as well, the cost function for moving from A to B was implemented as the weighted sum of 1) magnitude of delta-v, 2) straight line distance, and 3) time duration. The weights are specified as part of the scenario.

- The choice of cost function and the presence of moving obstacles necessitated the use of a grid in seven dimensional space. Each point is a partial robot body state, including three dimensions for position, three dimensions for incoming velocity, and one dimension for time. Another way to view this is that in space, the advantage of being at a certain position depends upon when you are there, since hazards are in motion, and what your velocity is, since turning and braking in space are expensive.

  A reference frame also had to be chosen for the search space. The NASA simulation employs an earth-centered inertial frame and has full frame transformation capabilities. AI path planning is usually terrain based or airborne and uses a local, non-inertial frame. For this work, a local vertical, local horizontal (LVLH) frame, on orbit and centered on the platform of interest, was selected. Orbital effects are easily modeled in this frame, and position and velocity vectors for obstacles and robot are of a scale which the grid search can manage.

- When spatial distance is the grid search cost function, the heuristic cost factor is usually the straight line distance from path endpoint to the goal. For the on-orbit algorithm, the

heuristic cost is computed by solving a two burn problem, one at the path endpoint and one at the goal, to reach the goal in a direct (but not necessarily straight line) trajectory.

- A uniform grid and a definition of the neighbors to each point was needed for the seven dimensional search space. In two spatial dimensions it is sufficient to specify a resolution distance d to achieve a uniform grid. All points (id, jd) for integer i and j are on the grid. A point's neighbors are usually declared to be any point on the grid less than 2d distance from the point (8 neighbors) or 1d distance away (4 neighbors). This is easily extended to three spatial dimensions. If it is extended to the fourth dimension of time by the declaration of a time resolution t, however, so that neighbors are, say, 1d distance away and 1t time away, the effect is to fix the speed of robot to the one value d/t. This is undesirable for an on-orbit robot.

  Many schemes are possible which result in a uniform grid in seven dimensions, a small number of neighbors for each point, and a variable speed robot. For the first experiment, a uniform grid was defined in four dimensions in terms of distance resolution d and time resolution t. A point was defined to be on a grid in seven dimensions if it was on the space/time grid and its velocity was one of the set of velocities achievable by arriving from a neighboring point. A point's neighbors were defined to be all those grid points in two sets:

  - those time t away and any distance away, subject to limits on maximum robot speed and maximum instantaneous delta-v magnitude, and
  - those points distance less than 2d away and any time away, subject to limits on minimum robot speed and maximum instantaneous delta-v magnitude.

  For example, assume t and d are 1, the robot is stopped at point (0, 0, 0) at time 0, and robot speed may vary from .4 to 2.5. Ignore limits on delta-v magnitude. Then the robot may move to (1, 0, 0) at times 1 or 2 (with speed 1 or .5), or to point (2, 0, 0) at time 1 (with speed 2), but not to point (1, 0, 0) at time 3 or point (3, 0, 0) at time 1, since robot speed limitations are exceeded. As another example, assume that robot speed is limited to be between 1 and 1.9. Then the neighbors of (0, 0, 0) at time 0 are the 26 points (i, j, k) at time 1, where i, j, and k are -1, 0, or 1, but not all 0.

5 Experimental Evaluation of the Algorithm

Table 2 summarizes the characteristics of a test scenario. A robot is directed to move to a point near one module of the space station for an inspection. An obstacle is moving by the station. This is unrealistic, but was included to test the ability of the robot to avoid moving obstacles. The robot has no velocity initially in the station-centered local frame. The goal points always have a zero desired velocity as well. The search occurs on a grid with 6 foot, 60 second resolution. The runs were arbitrarily limited to 6 hours of execution time.

The goal points all have a zero desired velocity and no specified arrival time. It was discovered that the algorithm can achieve a goal position and velocity at a reasonable time, if no arrival time is specified. This behavior is obtained by not testing for compliance with a goal time and by making a slight adjustment to the cost heuristic. In the first case, the two-burn problem for the cost heuristic consists of finding two delta-v's to move from a neighbor state (position, velocity, time) to a goal state (position, velocity, time). In the second case, the two-burn problem is modified to motion

Table 2: Characteristics of the Test Scenario

Objects	
Platform	Modeled after Space Station 14 Component objects: cylinders and rectahedrons
Obstacle	Cylinder, moving
Robot	Displayed as cylinder, modeled as sphere .09 feet/second minimum speed .19 feet/second maximum speed .2 feet/second maximum single delta-v magnitude 0 initial velocity 0 goal velocity
Scale	
Grid resolution	6 feet distance 60 seconds time

from a neighbor state (position, velocity, time) to a partial goal state (position, velocity). Since this is not sufficiently constrained to yield a single solution, the constraint is added that the goal time, for the one cost evaluation only, is set equal to the distance from neighbor to goal divided by the magnitude of the neighbor point's velocity. Effectively, the heuristic assumes that the robot will continue directly to the goal position at roughly its current speed.

Several experiments were performed to see how grid search would behave on the scenario for various goal points and cost function weighting schemes. The cost function for these runs was the weighted sum of the total delta-v and the total distance (time was not considered). The results are given in Tables 3 and 4. The path identified by the algorithm for the goal (8.667, -0.667, 9.833) and delta-v and distance both weighted 1 is illustrated in Figure 2. To summarize the results, the algorithm takes a very long time to find an obstacle free, fairly expensive trajectory.

To understand why the A* algorithm is taking so long, consider two points. First, a dynamic programming algorithm, without the cost heuristic, will not work in a domain where the robot may move from point to point without incurring cost. In space, the algorithm will establish a starting velocity for the robot, and will then head off forever in the direction of that velocity. It will have discovered a zero cost path of ever-increasing length, which it considers superior to all other paths requiring some delta-v. This does not happen in two spatial dimensions with a distance-based cost function, because it is not possible to move from point to point without incurring some cost (it is also true that dynamic programming in two dimensions is usually constrained to stay between the start and goal points on one dimension). The A* algorithm does not head off forever on a free trajectory. However, it still prefers to explore passive trajectories once initial velocity has been established, and the cost heuristic does not affect path evaluations enough to counter this effect until a great deal of exploration has occurred.

Second, for the on-orbit problem, the search space is very large. The seven dimension space and the desire for a variable speed robot lead to many neighbors at each point. A limit on the magnitude of the delta-v allowed at each point helps somewhat, but not enough.

Table 3: Number of Points Searched during Grid Search

Cost Function Weights: Delta-v Goal Distance	0.0 1.0	1.0 1.0	1.0 0.5	1.0 0.1	1.0 0.0
<1 2.167 0>	18	73	127	538	1883
<1 4 1>	67	274	492	2351	11,217
<8.667 -.667 9.833>	1560	33,361	*	*	*

* Indicates run did not complete within 6 hours.
The start, <0 0 0>, and goals are in grid-relative coordinates.

Table 4: Total Delta-V for Paths Identified by Grid Search

Cost Function Weights: Delta-v Goal Distance	0.0 1.0	1.0 1.0	1.0 0.5	1.0 0.1	1.0 0.0
<1 2.167 0>	0.355	0.355	0.355	0.333	0.333
<1 4 1>	0.485	0.420	0.420	0.420	0.420
<8.667 -.667 9.833>**	0.872	0.870	*	*	*

* Indicates run did not complete within 6 hours.
** For comparison, total delta-v for a three-burn, obstacle free trajectory was 0.404.
The start, <0 0 0>, and goals are in grid-relative coordinates.

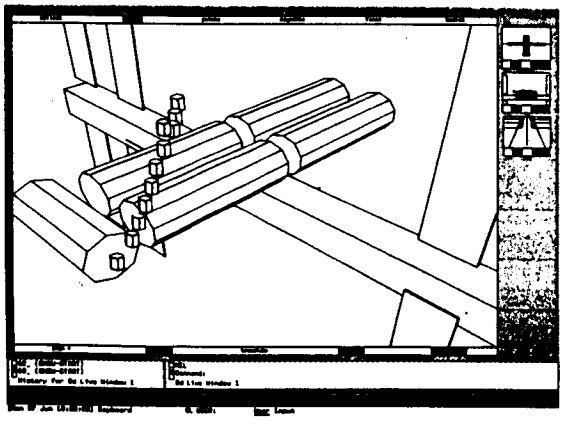


Figure 2: Path Identified by the Grid Search Algorithm

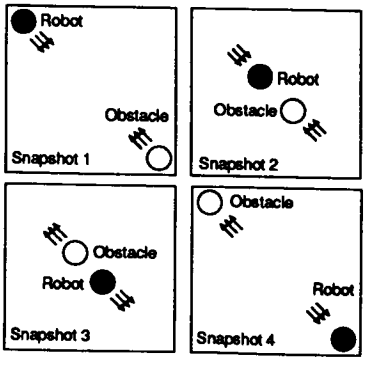


Figure 3: Approach to Obstacle Avoidance Permits Collisions on a Coarse Grid

The size of the search space in combination with the lack of tight focus on the goal makes the identification of paths of more than five burns unfeasible. In fact, the number of points evaluated grows exponentially with the number of waypoints in the path. The search problem is NP-hard, which means that order-of-magnitude improvements in computing hardware will only make a small dent in the performance problem.

However, guidance engineers have been quick to point out that the problem of Figure 2 could probably be solved with two or three burns. To verify this, a three burn, obstacle free trajectory was proposed by viewing the scenario and selecting an intermediate waypoint above the modules. Simulation of the trajectory revealed that its total delta-v would be half that of the trajectory identified by the A* search. 13 burns is a needlessly complicated and suboptimal solution.

This suggests that a coarser grid may be appropriate. Unfortunately, there is an upper limit to grid resolution which is imposed indirectly by the approach to obstacle avoidance. Obstacles are avoided by testing for robot-obstacle intersections at the states defined by the grid points. No consideration is given to whether the trajectories between grid points intersect obstacles. This is perfectly acceptable if the grid points are close together compared to the sizes and relative velocities of robot and obstacles. If the grid points are too far apart, the planner may promote a trajectory which appears safe at the waypoints, but which will in fact lead to a collision (see Figure 3).

Therefore, the conclusion from the first experiment was that a second algorithm was needed which would employ coarser grids and consider obstacle avoidance over the trajectories, not just at the grid points.

6 Proposed Second Experiment

The second experiment will utilize numerical techniques for collision testing and will assume that most trajectories are accomplished efficiently with a small number of burns. The approach is to solve a two burn problem for an optimal trajectory, and if the resulting trajectory fails to avoid obstacles, to move to successively more burns. The timing and location of additional burns will be identified by searching for burn points on a grid which is scaled to the current total number of burns. Thus the search will retain aspects of a grid search, but will occur on a sequence of grids of increasingly finer resolution.

Obstacle trajectories and robot trajectory segments between burns will be represented by polynomials. Collisions will be detected by testing the polynomials for intersection. Previous experiments indicate that obstacles which must be avoided in space can be modeled as circumscribing spheres and their entire vicinity avoided. Thus it should be possible to describe passive, spherical obstacles trajectories mathematically.

The platform of interest, on the other hand, must be modeled in more detail. Converting complex, moving shapes into mathematical representations is generally not feasible. However, by restricting the platform to be unarticulated and by employing a platform-relative reference frame, we can transform robot-platform collision testing into comparing a robot trajectory to a set of stationary shapes.

7 Conclusions

This project has demonstrated the potential for combining trajectory optimization and AI path planning for on-orbit robotic guidance. It has clarified several issues about algorithm design which require further investigation. In particular, it remains to be seen whether critical assumptions about the on-orbit scenario can be made which circumvent the combinatorial explosion in computation time. This explosion plagues all approaches to path optimization with obstacle avoidance.

This effort has also revealed several other directions for work in support of autonomous space robotics, including further analysis of requirements for servicing scenarios and development of functionality outside of predictive planning for free-flying guidance. In particular, research is needed on sensing, obstacle modeling, and reactive planning in conjunction with robot control.

References

[1] *Advanced Technology Advisory Committee, Executive Overview.* TM 87566, NASA, April 1984.

[2] *Space Station Automation and Robotics Plan.* Document D483-50055-1, Boeing Aerospace, October 1986. Data Requirement 17 for Work Package 01, NASA Contract NAS8-36526.

[3] A.F. Brindle, W. Kohn, G.M. Lobdell, and J.H. Albert. *Space Robot Path Planning: Project Proposal.* Document D180-30735-1, Boeing Aerospace, December 1987.

[4] H.H. Chen, N. Ahuja, and T.S. Huang. Septree representations of moving objects using hexagonal cylindrical decomposition. *Optical Engineering*, 23(5):531–535, September/October 1984.

[5] A. Elfes. A sonar-based mapping and navigation system. In *International Conference on Robotics and Automation, Volume 2*, pages 1151–1156, IEEE, San Francisco, CA, April 1986.

[6] M. Herman. Fast, three-dimensional, collision-free motion planning. In *International Conference on Robotics and Automation, Volume 2*, pages 1056–1063, IEEE, San Francisco, CA, April 1986.

[7] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, February 1983.

[8] C.E. Thorpe. Path relaxation: path planning for a mobile robot. In *Autonomous Mobile Robots Annual Report 1985, CMU-RI-TR-86-4*, pages 39–42, Carnegie-Mellon University, February 1985.

[9] P.H. Winston. *Artificial Intelligence.* Addison-Wesley, 1984.

Telepresence System Development for Application to the Control of Remote Robotic Systems

Carl D. Crane III, Joseph Duffy
Rajul Vora, Shih-Chien Chiang

Center for Intelligent Machines and Robotics
University of Florida
Gainesville, Florida 32611

I. ABSTRACT

This paper describes the recent developments of techniques which assist an operator in the control of remote robotic systems. In particular, applications are aimed at two specific scenarios: (1) the control of remote robot manipulators; and (2) motion planning for remote transporter vehicles. Common to both applications is the use of realistic computer graphics images which provide the operator with pertinent information. This paper will describe the specific system developments for several recently completed and ongoing telepresence research projects in the Center for Intelligent Machines and Robotics (CIMAR) at the University of Florida.

II. INTRODUCTION

Significant advances have been made in a broad spectrum of component technologies such as kinematics and dynamics, control, vision and pattern recognition, obstacle avoidance, computer graphics, and artificial intelligence. Each of these technologies can individually impact on an operator's efficiency in the control of a remote manipulator or transporter. An objective of telepresence systems is to combine these component technologies in order to provide the operator with multiple sensory feedback data. In this manner an operator can in effect directly experience the environment of the remote manipulator system. The abundance of sensory feedback coupled with computer assisted operation (a low level of autonomy) is what distinguishes telepresence from earlier teleoperated systems.

The operational scenario of remote systems may vary significantly from case to case. In some situations, the operator may be required to control the remote system in real time for which it is necessary for the remote system to respond directly to operator input. In many circumstances, however, a real time response is not possible or desired. A significant communication time delay, for example, can make real time control awkward at best[1]*. A second classification can be made regarding the operational environment. Clearly, it is necessary to know whether or not the operational environment of the remote system is known a priori. Knowledge of an accurate model of the environment can significantly impact the type of telepresence system which is to be implemented. Operation in a structured environment such as a nuclear power plant [2] is considerably different from controlling a remote transporter in the field.

*Numbers in brackets refer to references at end of paper.

This paper will describe several projects conducted at CIMAR which all deal with the control of remote manipulator or transporter systems. For each case, a scenario is established where: (1) the operator is either directly controlling the system in real time or is performing an off-line planning task; and (2) the environmental descriptive model is known a priori or the system is operating in a totally unknown area.

III. CONTROL OF REMOTE ROBOT MANIPULATORS

A. Initial Telepresence System [3]

The first telepresence system developed at CIMAR was concerned with the real time control of a remote manipulator in an unstructured environment. It was assumed that the operator could not directly see the manipulator or the objects in its workspace as he was directly maneuvering the robot with a joystick device. The task for the operator was to acquire and move several cubes in the workspace without coming into contact with a sphere.

The primary sensory feedback required to perform this task is vision. In many instances, vision alone would give the operator sufficient information in order to manipulate the cubes without contacting the sphere. Logically, this vision could be provided by video cameras which surround the workspace of the manipulator or which are attached to the manipulator itself. It should be noted, however, that the use of video cameras to provide vision feedback does have certain disadvantages. Primarily, more than one camera must be used to provide sufficient viewing positions. Each of these cameras will most likely have at least three degrees of freedom (zoom, tilt, and pan) thereby giving the operator numerous additional parameters to control. Furthermore, environmental conditions may cause the video image to be blurred or poor lighting and contrast may make the image confusing and unclear.

Because of these limitations, the goal of this project was to evaluate the operator's performance when the vision feedback was provided via a computer graphics display. The use of computer graphics offers three distinct advantages. First, images are clear and sharp since colors and contrast can be selected and programmed. Second, the computer graphics system allows for the viewing of the image from any desired vantage point. This ability removes the requirement for a multitude of video cameras and allows the operator to focus attention on only one monitor screen. Third, the computer graphics system can provide additional feedback to the operator concerning imminent collisions. When the manipulator was moved close to the spherical obstacle, the color of the sphere was changed and an audio tone was emitted. This additional feedback regarding collisions significantly improved operator confidence and performance. The collision warning feature was implemented by placing the obstacle in an imaginary rectangular protective box. The graphics system in effect compares the image of the robot with this protective box and determines in real time whether any part of the robot is in contact.

A photograph of the initial telepresence system is shown in Figure 1 while the system configuration is shown in Figure 2. The system consisted of the following four component technologies:

(1) Robot manipulator.
(2) Six degree of freedom universal joystick.
(3) Obstacle detection vision system.
(4) Computer graphics system.

The manipulator was drawn on the graphics screen by communicating the joint angles from the manipulator to the graphics workstation at every instant. The vision system was used to locate the objects in the manipulator workspace so that they could be displayed on the graphics screen. Shown in Figure 3 is an image of the robot as it comes close to the obstacle.

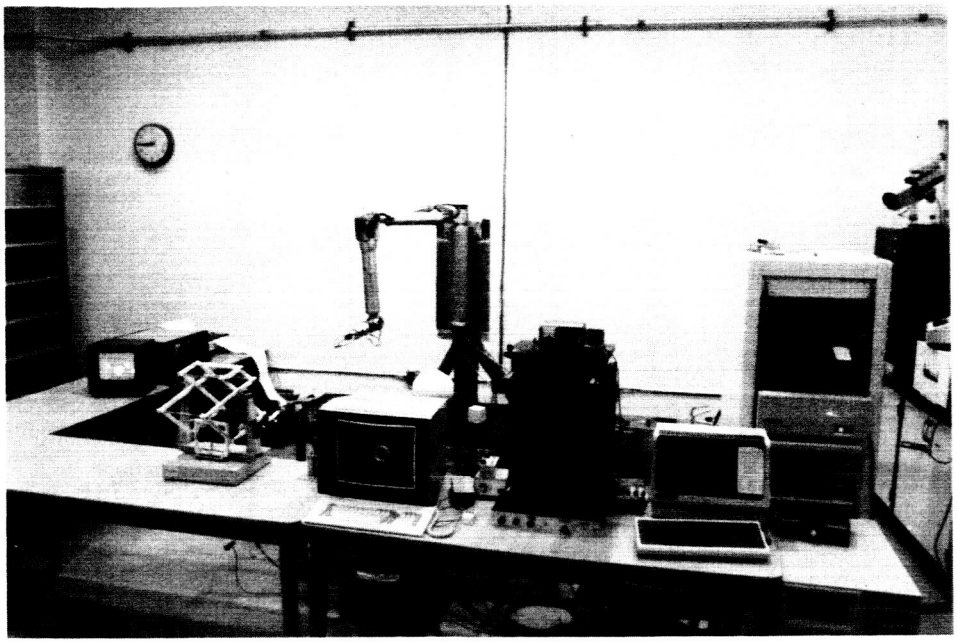


Figure 1: Telepresence System

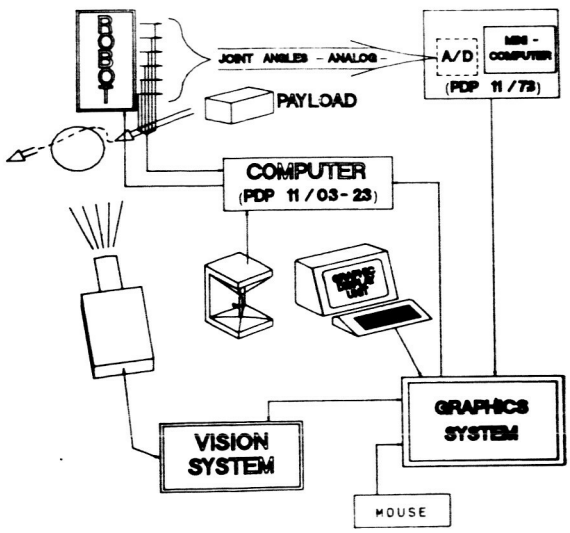


Figure 2: System Configuration

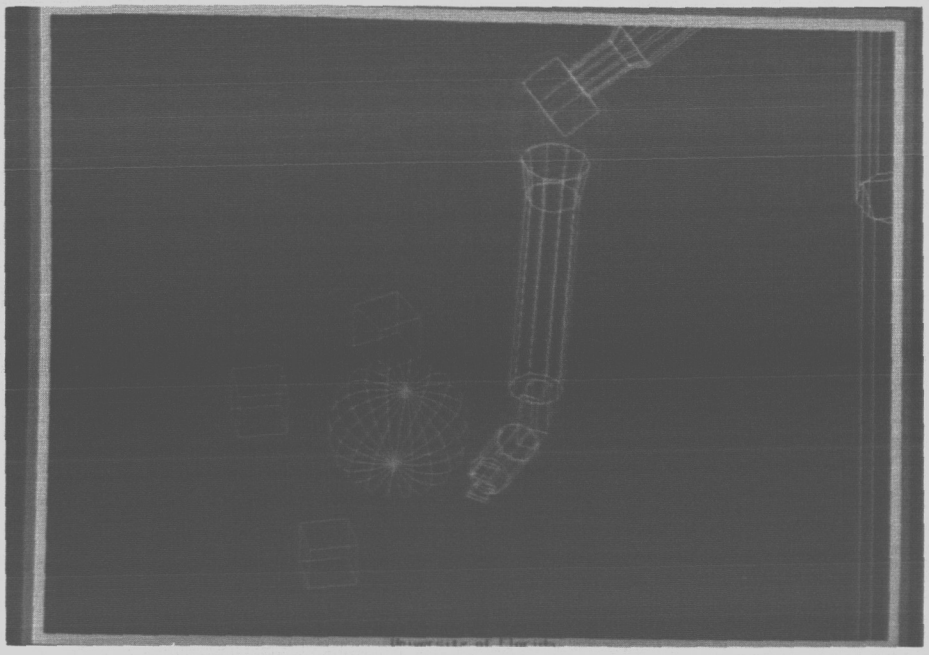


Figure 3: Warning of Imminent Collision

The initial telepresence system showed that it was feasible and practical to utilize a computer graphics display to control a remote robot manipulator in real time in an unknown environment. The clarity of the image coupled with the real time determination of imminent collisions with obstacles significantly improved operator performance.

B. Robot Path Processor [4]

As an extension to the previous project, an investigation of off-line task planning was conducted. For this project it was assumed that the location of objects in the workspace of the robot were known (either from a priori knowledge of a world model or from a sensor scan of the workspace). Off-line task planning offers an advantage if communication delays prohibit real time operation, or if an operator can plan a task much quicker than the manipulator can execute it. Further, off-line planning offers the operator the option to edit or modify the task before it is sent to the manipulator for execution.

During this project, emphasis was placed on the development of the task editor. The objective was to be able to allow the operator to rapidly enter a robot task and then be able to refine it. The resulting system consists of the same universal joystick controller and computer graphics workstation used in the previous project together with a PUMA 560 industrial robot. The operator moves the animated robot on the screen and then can go back and perform such tasks as moving specific points on the path or modifying certain sections to be perfectly straight.

One problem that had to be overcome, however, was how to move the joystick in order to get a desired motion of the robot on the graphics screen. Confusion resulted in that the operator could look at the robot from any viewing position and then had to mentally reference the joystick and manipulator coordinate systems (see Figure 4) . This problem was overcome by introducing an additional coordinate transformation which mapped the joystick coordinates into the screen coordinates. The result was that a movement of the joystick to the right or left would cause the end effector of the robot to also move to the right or left of the graphics screen, and so on. In this

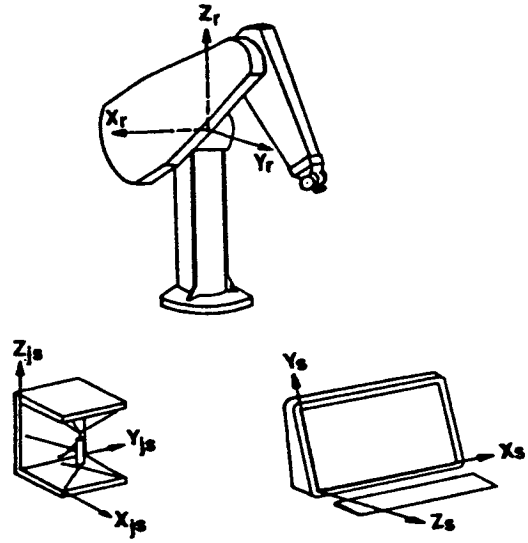


Figure 4: Coordinate System Reference Frames

manner, the operator could always correlate the motion of the robot in response to any joystick move, no matter how the scene was being viewed. This feature significantly improved the off-line planning process and was also later applied to the real time control scenario.

C. Off-Line Programming with Autonomy [5]

Off-line task planning can be significantly enhanced by the introduction of autonomy. In this project it was again assumed that the location of objects in the workspace of the robot were known, either from a priori knowledge of a world model or from a sensor scan of the workspace. The objective of this effort was to develop algorithms which would autonomously generate robot motions and tasks based on high level user inputs.

The development of motion planning algorithms which avoid obstacles was a major focus of this effort. Shown in Figure 5 is a path which has been generated in order to move a manipulator between two user specified positions. A significant feature of the algorithm is that resulting paths are generated rapidly (under five seconds of computation time) [6]. The algorithms take advantage of the knowledge of the geometry of the manipulator and its environment in order to avoid excessive heuristic searches.

Common to all off-line approaches, the operator is given the opportunity to preview the planned task on the graphics screen prior to the manipulator beginning the task. In this manner, the operator can evaluate and critique the performance of the autonomous algorithms.

This research program was conducted for the U.S. Army Belvoir R,D,&E Center, Fort Belvoir, Virginia. The specific application for this project involved logistics operations and is shown in Figure 6. A camera on board the fork attachment was able to locate a target mounted on a pallet and then calculate the position and orientation of the pallet relative to the camera. With this information, the operator would be able to acquire the pallet and maneuver it off-line via the graphics animation system. Once a task was planned, previewed, and approved, the manipulator would be commanded to perform the operation.

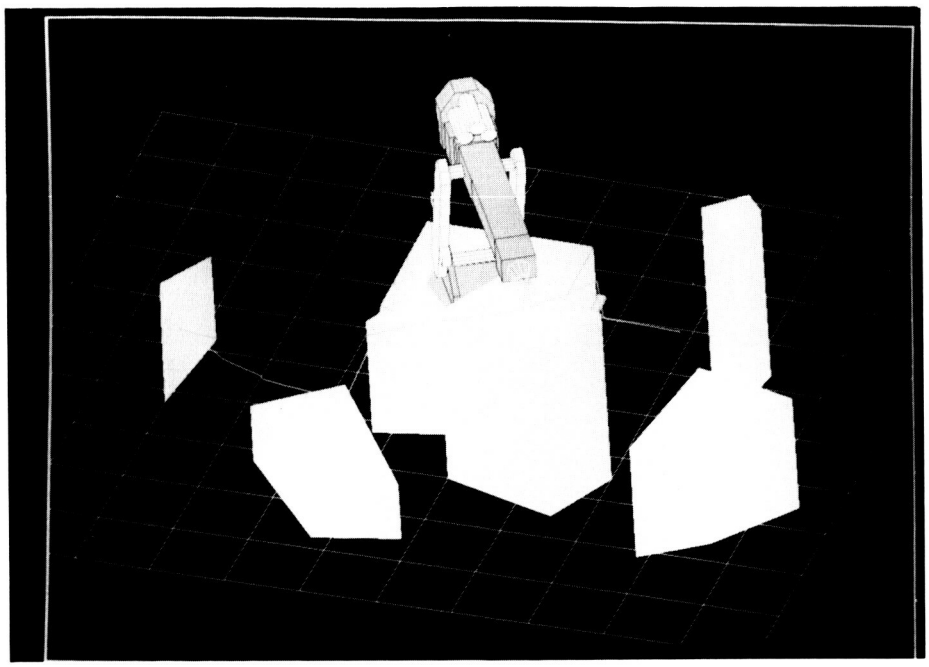


Figure 5: Autonomously Generated Path

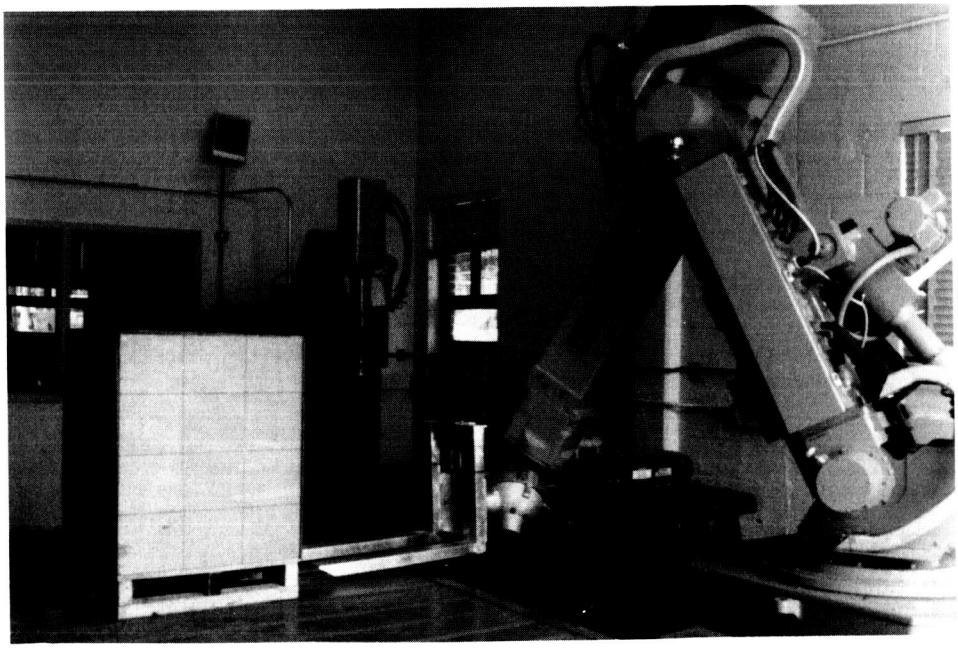


Figure 6: Logistics Application

IV. MOTION PLANNING FOR REMOTE TRANSPORTER VEHICLES

Many of the man/machine interface control issues associated with the operation of remote transporter systems are directly analogous to those already discussed for manipulator systems. In particular, the transporter operation can be categorized according to whether or not it moves in real time under operator control and as to whether or not a prior knowledge of the environment is known.

The work in this area is sponsored by the U.S. Department of Energy. The University of Florida, in cooperation with the Oak Ridge National Laboratory and the Universities of Michigan, Tennessee, and Texas is developing an advanced robotic system under the University Program for Robotics for Advanced Reactors. As such, the developments in this area are specifically aimed at the operation of a transporter vehicle in a nuclear power plant environment where it is assumed that an accurate world descriptive model exists.

A. Development of an Articulated Transporter/Manipulator System (ATMS) [7]

The complex obstacle strewn environment of a nuclear power plant requires that the robotic system be capable of crossing or jumping over substantially sized obstacles. Transport mechanisms consisting of combinations of wheels, tracks, and legs were considered as candidates, but the inherent mobility limitations of these mechanisms led to the selection of a transporter comprised of multiple articulated segments (see Figure 7).

The ATMS is comprised of eighteen individual segments which provide both maneuverability and locomotion. Each segment is twenty four inches in length, and has a pair of motor driven wheels to provide traction for forward and reverse motion. Segments are connected in series by a pair of revolute joint axes. The significant feature of the design is that the ATMS will be able to cross over horizontal gaps of up to twelve feet in length.

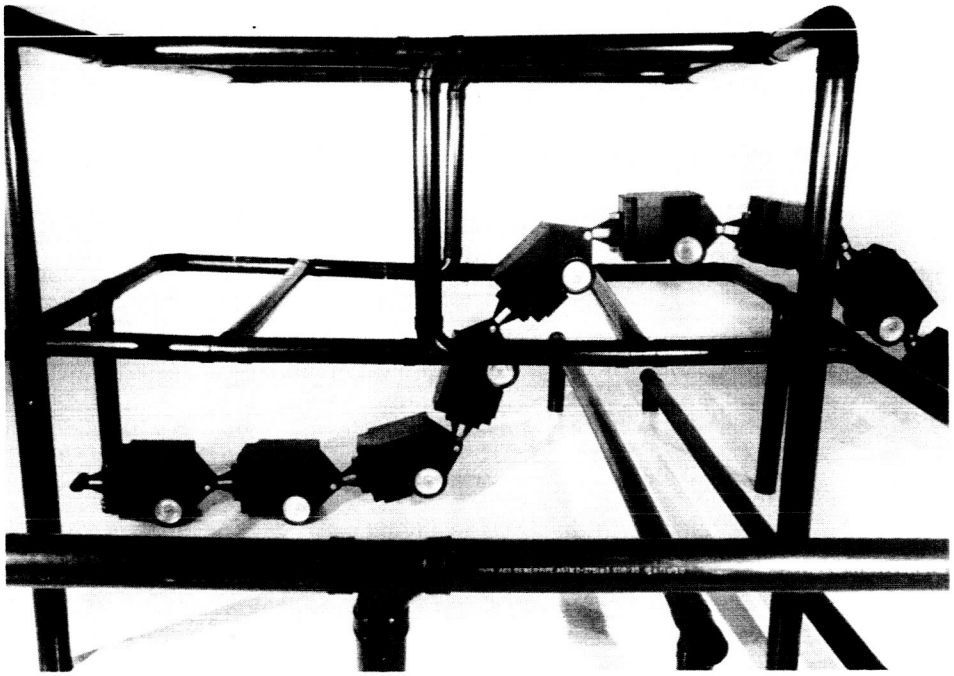


Figure 7: ATMS Model

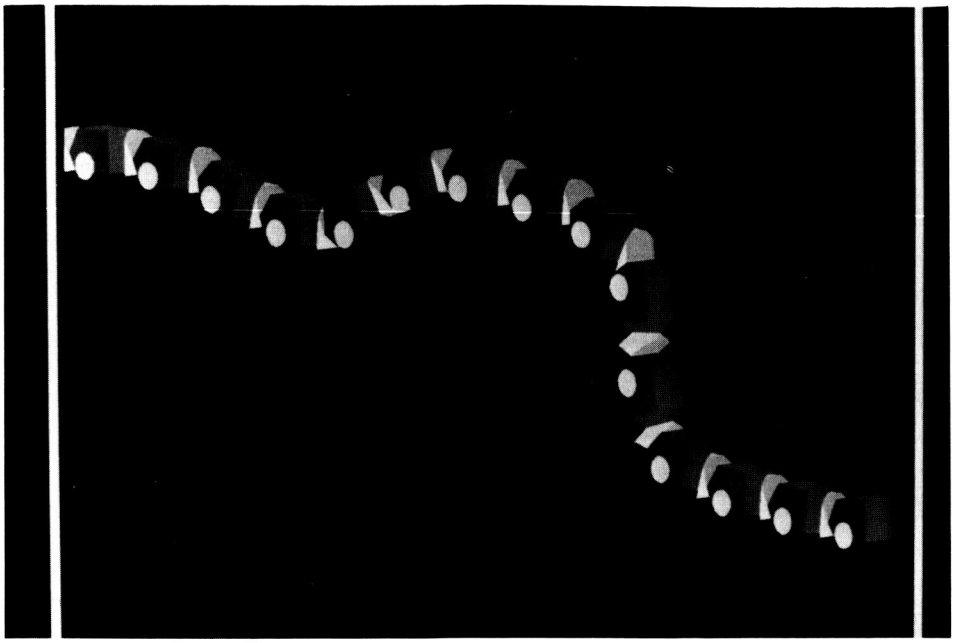


Figure 8: Animated Representation of ATMS

Initial control of the ATMS has been established by having an operator specify the direction and velocity of the lead segment. Subsequent segments of the ATMS will follow the path of the segment directly in front of it (see Figure 8). Work to date includes the development of follow-the-leader algorithms for vertical and horizontal navigation. Further, a series of stability rules have been generated which govern the effective maneuverability of the system. During a man-controlled real time operation of the system, the stability rules will be continuously evaluated and monitored, and the operator will be warned of potentially unstable actions. This combination of direct man-controlled operation with computer assistance is a fundamental principle of the telepresence concept. In addition to the real time human control, off-line autonomous control algorithms are currently being implemented.

B. HERMIES Implementation

A fully operational version of the ATMS has yet to be fabricated. In order to demonstrate the progress of the research program, the HERMIES robotic transporter at the Oak Ridge National Laboratory is being used as a system demonstrator (see Figure 9). HERMIES is limited to performing planar motion. However, many of the man-machine control issues which will involve the ATMS can be resolved via HERMIES.

The system demonstration combines off-line and real time operation. During the planning phase, the operator can plan a motion path via (1) the computer graphics system alone (Univ. of Florida), (2) a three degree of freedom force reflecting joystick (Univ. of Texas), or (3) an autonomous path planning algorithm (Univ. of Michigan). Once a path is planned, the operator is able to review it and modify it if necessary prior to HERMIES beginning its motion. Figure 10 shows the animated display provided to the user when a path is being planned. Three interactive windows are provided which present a top view, the view from HERMIES, and the view from a user controlled camera located anywhere in the environment.

An important aspect of the planning phase is the time acceleration concept[8]. The operator can effectively specify a time scale factor so that the planning phase can be accomplished in an accelerated time scale. For example, the ATMS can achieve a maximum velocity of only two feet per second. By selecting an accelerated time

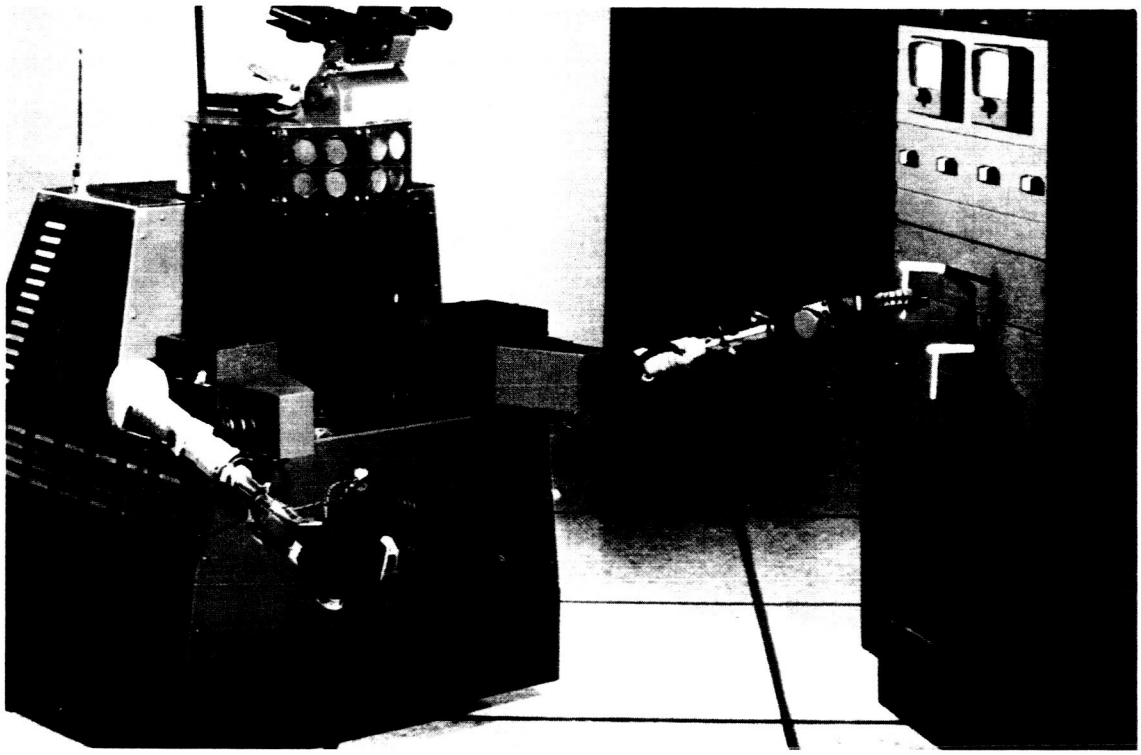


Figure 9: HERMIES Mobile Robot

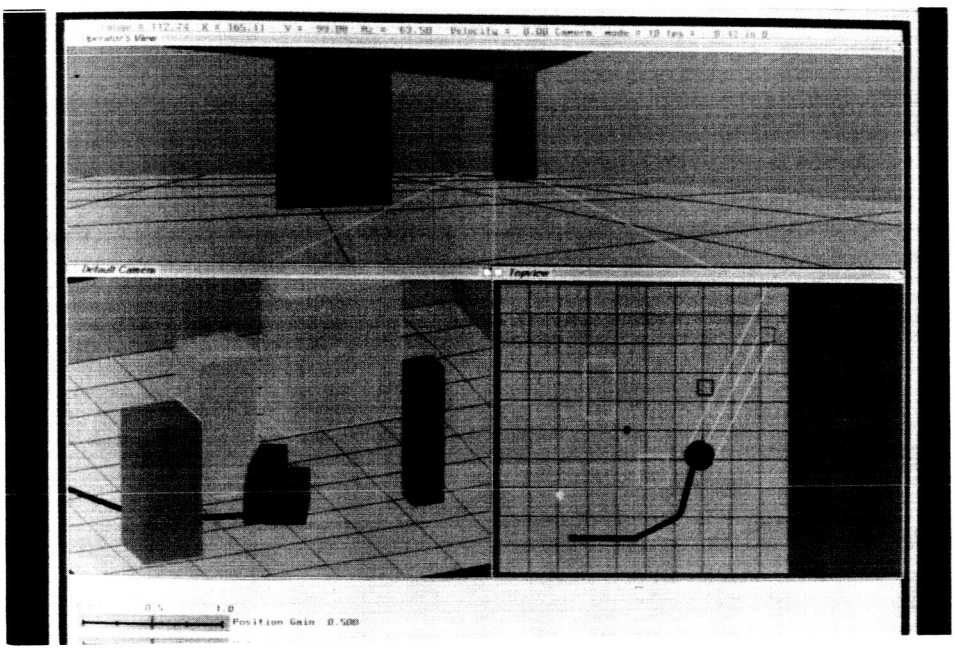


Figure 10: Operator's Console Image

scale, the operator will be able to perform the motion planning as if the ATMS had an unlimited maximum velocity. Upon the actual motion of the ATMS, the plan would be reconstructed in the real time domain.

V. CONCLUSION

Several recent and ongoing research programs have been described which all deal with some aspect of telepresence system development. Varying techniques have been discussed which assist an operator in the control of remote robotic systems either in real time or off-line, or when a prior knowledge of the environment is known or not.

It appears evident that any resulting telepresence system must be able to perform in all of the operational categories. For example, although a nuclear power plant may be a structured environment for navigation purposes, the environment can become unstructured during a manipulation task. Because of this, the generic techniques that were developed here can be combined together in order to offer the operator a highly sophisticated man-machine interface.

VI. ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions made by other current and former members of the Center for Intelligent Machines and Robotics, Dr. R. Harrell, Dr. J. Staudhammer, Mr. Y. Choi, and Mr. V. Chesney. Significant contributions were also made by Mr. M. Locke of the U.S. Army Belvoir R,D,&E Center. Further the authors wish to acknowledge the financial support of the U.S. Department of Energy, U.S. Army Belvoir R,D,&E Center, McDonnell Douglas Astronautics Company, and of the Military Avionics Division of Honeywell.

VII. REFERENCES

[1] L. Conway, et al., "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology," 1987 IEEE International Conference on Robotics and Automation, Raleigh, N.C.

[2] J.S. Tulenko, et al., "An Advanced Semi-Autonomous Robotic System for Hazardous Response Work at Nuclear Power Stations," Proceedings of the Winter Meeting of the ANS, Los Angeles, Ca., January 1987.

[3] C.D. Crane and J. Duffy, "An Interactive Animated Display of Man-Controlled and Autonomous Robots," Proceedings of 1986 ASME Computers in Engineering Conference, Chicago, July 1986.

[4] C.D. Crane, Y. Choi, G. Matthew, "Interactive Off-Line Robot Path Processor," Proceedings of the 1988 ASME Computers in Engineering Conference, San Francisco, July 1988.

[5] C.D. Crane, J. Duffy, and M. Locke, "Off-Line Programming and Path Generation for Robot Manipulators," Intelligent Systems for Army Logistics Support and Combat Engineering Symposium, Ft. Belvoir, Va., 25-27 March 1986.

[6] C.S. Tseng, "Rapid Generation of Collision-Free Paths for Robot Manipulators with Computer Graphics Animation," Ph.D. Dissertation, University of Florida 1987.

[7] C.D. Crane, S. Chiang, et al., "Algorithm Development and Computer Graphics Simulation of an Articulated Transporter/Manipulator System," Proceedings of the 1988 ASME Computers in Engineering Conference, San Francisco, July 1988.

[8] C.D. Crane, J. Tulenko, et al., "Faster Than Real Time Robot Simulation for Plan Development and Robot Safety," submitted for publication, 1989 IEEE International Conference on Robotics and Automation.

ROBOTIC VISION

3D MODEL CONTROL OF IMAGE PROCESSING

An H. Nguyen and Lawrence Stark

Telerobotics Unit

University of California at Berkeley

INTRODUCTION

Telerobotics studies remote control of distant robots by a human operator using supervisory or direct control. Even if the robotic manipulator has vision or other senses, problems arise involving control, communications, and delay [18]. The communication delays that may be expected with telerobots working in space stations while being controlled from an Earth laboratory have led to a number of experiments attempting to circumvent the problem (Fig. 1). This delay in communication is a main motivating factor in moving from well-understood instantaneous hands-on manual control to less well-understood supervisory control [5,7]; the ultimate step would be the realization of a fully autonomous robot.

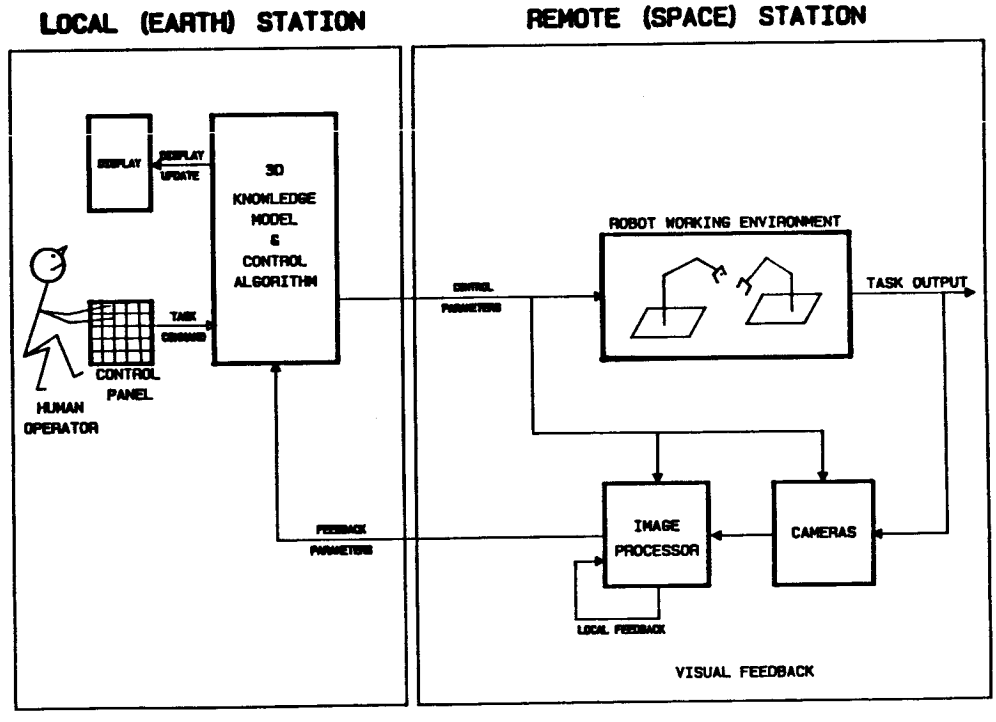


Fig. 1. Overview sketch of model control of robot working environment and of image processing

METHODS

Hardware Setup: Two a-robots (Armatron robots), modified to interface with an AT-386 computer [13,20] via parallel I/O ports, are controlled by computer in an autonomous mode. Manual control capability is preserved for teaching and for the supervisory mode, since hands-on control is vital in developing and evaluating different control algorithms. Both a-robots operate within a one-cubic-meter working environment. One of the robots (painted dark blue) is fitted with ONSNEs features (Fig. 2, left panel). Three cameras were used; one, an inexpensive C-mounted TV camera (Panasonic, model WV-1410); the others, commercially available 8mm camcorders (Sanyo, model VM-10). They provided two orthogonal side and top views, and an oblique view (Fig. 2, left panel). The computer selected among the camera views by means of a four-channel video multiplexer, whose output was connected to a simple frame grabber (Epix-Silicon Video, Chicago). The frame grabber resided in the AT bus and was directly controlled by the computer to digitize video images into 320 x 240 arrays of 8-bit pixels.

Software: Besides the main program performing administrative work, three major pieces of software were developed to control the mobile a-robot in obtaining a given target with visual feedback. These consisted of the ICM, 3DM, and utility programs. The ICM program included many different low-level image processing algorithms such as edge enhancement, feature extraction, automatic thresholding, filtering, moments computation. The 3DM program supported a complete, scaled-down model of the a-robot and its RWE. It also provided 2D projections of different camera views, and contained an algorithm for simple path planning. The utility software was highly optimized, and consisted of all the primitive functions for the frame grabber, EGA graphic display and plotter. All software was written in "C".

RESULTS: THE 3D MODEL

At the local earth station, the human operator views a display of the 3D model and uses the control panel in a supervisory mode to oversee the control algorithms (Fig. 1). At the remote space station, the control parameters drive the robots in the robot working environment (RWE). These control parameters also drive the cameras and the image processing algorithms. Besides a local feedback process, the main feedback is from the remote image processing to the Earth station 3D model.

A remote RWE is modeled using graphics workstation (Iris) with 3D graphic transformation support hardware (Fig. 3). At the RWE, three a-robots perform tasks; the m-robot (Mitsubishi manipulator) holds a camera and actively searches for optimum views. This experimental set-up provides us with a global view of the telerobotics control situation wherein several robots cooperate in a joint task, or each robot has an individual task assigned (Fig. 3). The 3D model is constructed from information about the robotic manipulators, the work pieces, and the camera positions [17]. The 3D model guides the image processor in extracting information derived from regions-of-interest (ROI) which contain on-the-scene visual enhancements (OSCNE) (Fig. 2, left); note the model with on-the-screen visual enhancements (OSCRN) [10,11] for use by the human operator (Fig. 2, right).

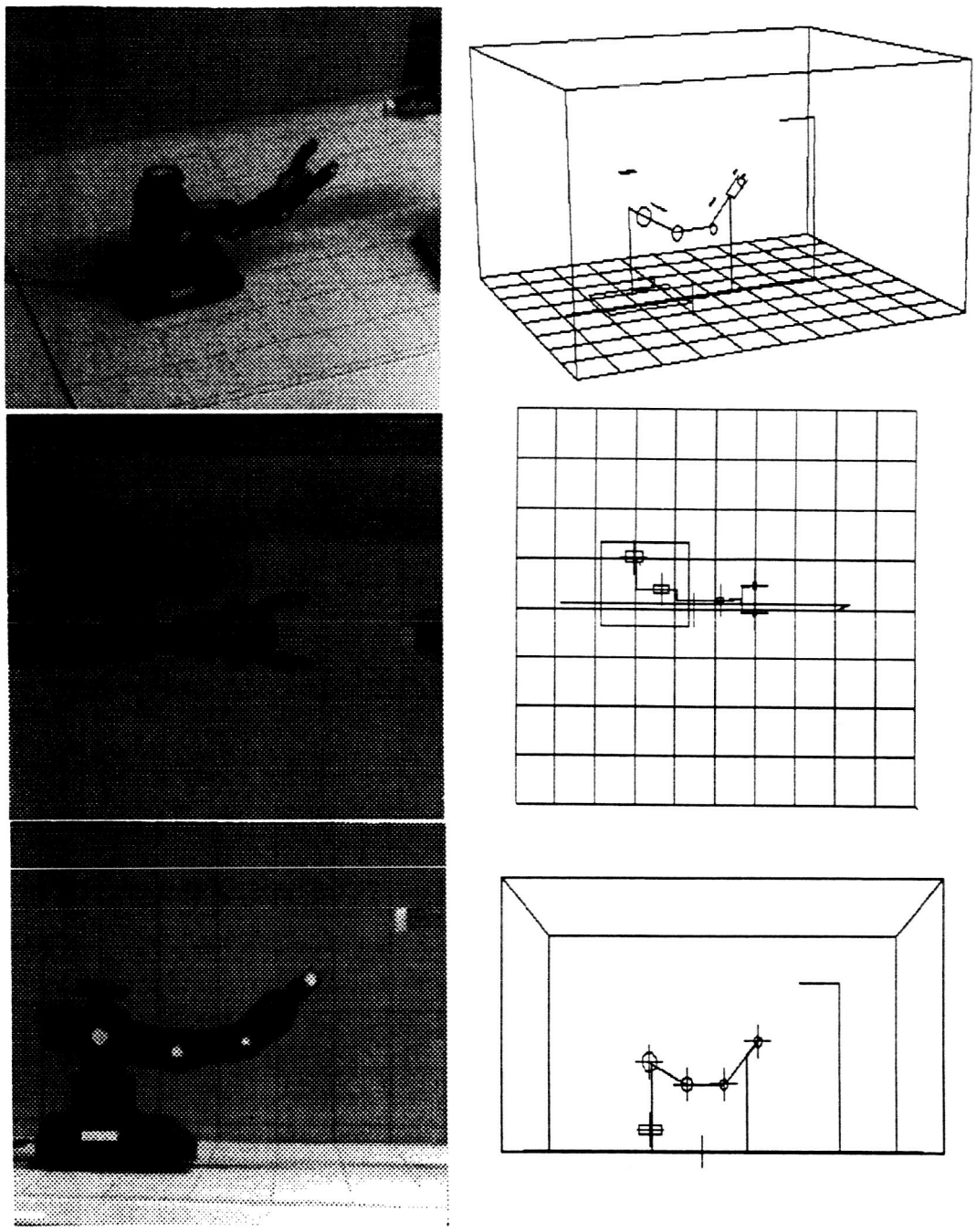


Fig. 2 Perspective views and orthogonal projections of a-robot (left) and model (right) showing on-the-scene visual enhancements (OSCNE). 3D model guides image processor to extract information only in regions-of-interest.

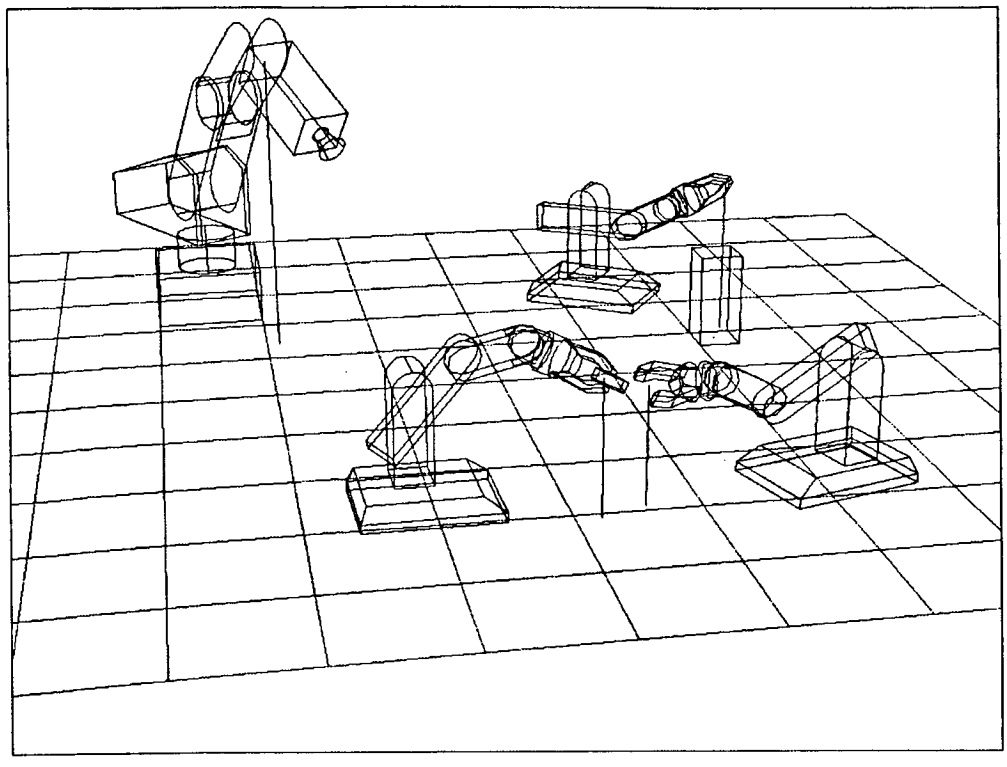


Fig. 3. Perspective view of vectorgraphic model of three a-robots and one m-robot.

Feature_selection_criteria_and_ROIs_locations: An important step in using top-down image processing to control robots is to specify the most useful information to be gained from the robot and its environment, and to determine the physical location of these features. These selections strongly influence not only the choice of information processing strategies, but also the image processing schemes employed.

Supposing that links of a robot are known or fixed, the kinematic recovery of the 3D robot model simply requires the joint location information. Any two consecutive joints of a robot provide complete information about the link length and orientation. In situations where the robot joint is not visible to the system, link orientation becomes important. In complex, multiple-robot environments, the image processing computer faces the far more challenging problems of occlusions, light reflections, shadows, noises, etc. Although the model uses a priori knowledge that plays an active role in resolving many of these problems, image processing tasks can be further simplified by introducing ONSNEs to both robots and the RWE (Fig. 4). These ONSNEs boost video signal to noise ratios within ROIs, and also may provide redundant information depending on their sizes and shapes.

Assignment_of_ROIs_locations: Each orthogonal projection view of the robot and its RWE has two sets of ROIs, the primary and secondary sets (Fig. 4). For the side view, the primary set (Fig. 4, upper) of ROIs is responsible for information about robot joints, while the secondary set of ROIs determines the robot orientations (Fig. 4, lower). Under static conditions, sizes of ROIs depend on those of ONSNEs. Since processing time is directly proportional to ROIs areas, ROIs should be small to minimize processing time, yet large enough to cover individual ONSNEs within ROIs. For automatic thresholding, optimum-size ROIs areas would be twice that of ONSNEs.

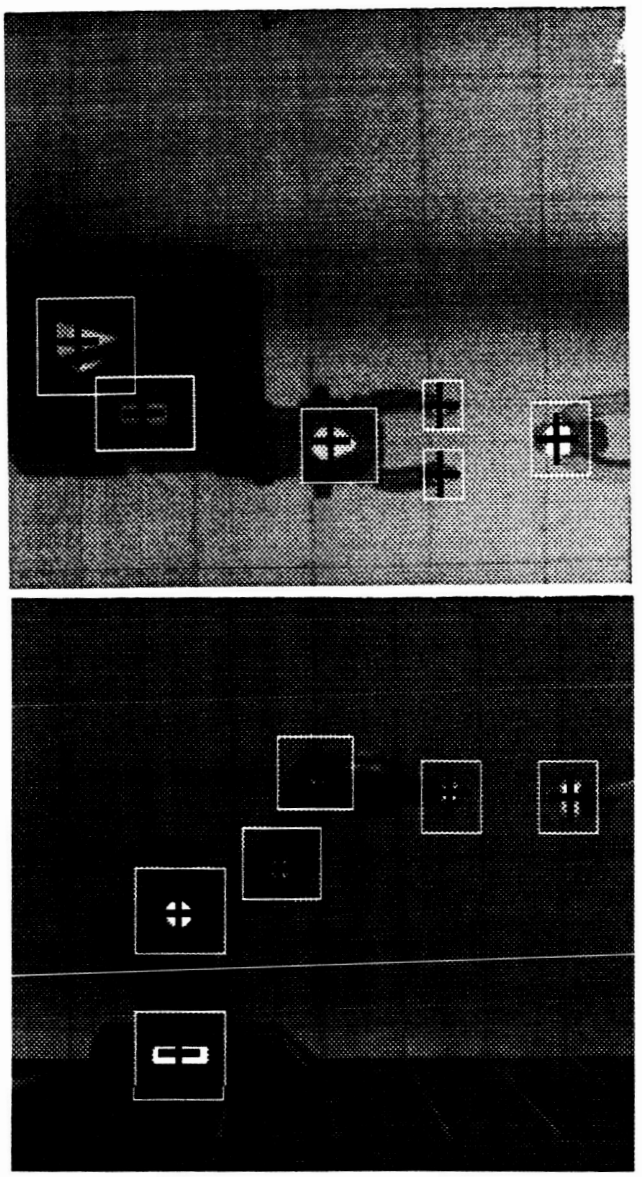


Fig. 4 Orthogonal views of actual robot reaching a target. While centroids of OSNCEs, resided within ROIs, provide feedback information for model to guide robot to target.

RESULTS : 2D IMAGE PROCESSING

Image processing operating within ROIs: As mentioned earlier, selected features determined both the locations of ROIs and the image processing schemes within them [13,19]. For instance, detecting angles of a robot link includes two steps: edge enhancement and line detection. The edge enhancement operation accentuates edges and acts like a two-dimensional high pass filter. Edge detection has been an active area of research for many years, and this continues today. Several algorithms for edge detection, such as Sobel, Kirsch, Roberts, and the Laplacian [6,8,14,16], are available and already implemented in VLSI devices [15]. These are simple operators in the form of 3 x 3 matrices. Another edge detection algorithm also worth mentioning is the Laplacian of the Gaussian [12]. This algorithm detects local edges effectively, and has been proven to be an optimum operator in dealing with true edges and noisy images [3]. Unfortunately, this operator requires a much larger kernel and, therefore, is computationally expensive. Since Sobel operators operate in pairs (the x and y directions independently), noise tends to be suppressed in one direction, while edges are accentuated in the other direction [15]. Due to their insensitivity to noises, simplicity in implementation, and efficiency in operation, the Sobel operators were incorporated into our scheme for low-level image processing in detecting edges.

Enhanced edges, resulting after Sobel operation, contain much higher intensity levels than the average. Therefore, appropriate threshold levels can be easily found, either by manual selection aided by histogram displays, or automatically by a thresholding algorithm. Threshold operations transform a gray level image into a binary image with two levels of intensity. Only enhanced edges above the threshold level remain after thresholding and are then ready for line detection. Orientation of a line can be retrieved by a number of algorithms such as matched filters, cross-correlation, or the Hough transform. Among these techniques, the Hough transform combined with top-down information from the model renders line information quite reliably and efficiently.

Centroid moment processing for the primary set of ROIs: Visual information residing in the primary set of ROIs provides sufficient feedback information for model adjustment and correction. Image processing carried out for this set of ROIs takes precedence over many other tasks, including control of the robots (Fig. 3, right panel; Fig. 4). Because of the strategic importance of this critical joint information, the ONSNEs were introduced (Fig. 3). The ONSNEs yield higher contrast in the video images, and thus more reliable visual information can be obtained under various luminance conditions.

Centroid and other invariant moments: The ONSNEs also have had a strong influence on the selection of the low-level image processing < 3 scheme used --- the invariant moments, a method in which centroids are derived. This technique had been previously applied to pattern recognition for printed characters [1,9], to chromosome analysis [2,4], and biological instrumentation [20]. The first three order moments yield information about size, centroid location, and major axis orientation for a bounded object; they are simple in implementation and inexpensive in computation. Additional higher order moments are also available for shape description, features that cannot be acquired from other low-level imaging schemes. Furthermore, the centroid parameters provide excellent information for local feedback (see Fig. 1), a special requirement for our image processing scheme.

Moments are widely utilized in classical mechanics; moments of a distribution function are also commonly used in statistical theory. For a given bounded, two-dimensional function f(x,y), the set of moments is defined as

$$M_{i,j} = \iint x^i y^j f(x,y)\, dx\, dy, \qquad i, j = 0, 1, 2, \ldots \tag{1}$$

In the infinte set [Mi,j] moments, as i and j take all non-negative values, uniquely determining the function f(x,y); and conversely, f(x,y) uniquely determines the set [Mi,j]. [i+j] is the order of the moment.

For binary images in which intensity of the object bounded by f(x,y) is one and zero elsewhere, the zeroth order moment M00

$$M_{00} = \iint f(x,y)\, dx\, dy \tag{2}$$

is the area of the object. Coordinates of the centroid are found to be,

$$\begin{aligned} xc &= M_{10}/M_{00} \\ yc &= M_{01}/M_{00} \end{aligned} \tag{3}$$

where M10 and M01 are the first moments for x and y respectively. Moments computed after translation of the origin to the center of gravity, are called central moments,

$$m_{ij} = \iint (x-xc)^i\, (y-yc)^j\, f(x,y)\, dx\, dy \tag{4}$$

For digitial image processing, equations (1) and (4) above, become

$$M_{ij} = \sum_x \sum_y x^i y^j\, f(x,y) \tag{5}$$

$$m_{ij} = \sum_x \sum_y (x-x_c)^i\, (y-y_c)^j\, f(x,y) \tag{6}$$

The second order central moments are [6]

$$m_{11} = M_{11} - y_C\, M_{10} \tag{7.a}$$

$$m_{20} = M_{20} - x_C\, M10 \tag{7.b}$$

$$m_{02} = M_{02} - y_C\, M01 \tag{7.c}$$

The object orientation or principal axis of rotation about this axis causes the second-order central moments to vanish [8,9]

$$\text{theta} = (\tan^{-1}\, (2m_{11}/(m_{20}-m_{02}))\, /\, 2 \tag{8}$$

All the area-normalized central moments relative to this principle axis are invariant under magnification, location, and rotation of the object [6,9].

RESULTS: AUTONOMOUS CONTROL

Control robot sequence: There are a number of different paths via which the robot can reach the target. However, for the purpose of this study, we derived a simple but effective algorithm to enable the 3D model to control the robot and to direct the image processing computer. The scheme worked satisfactorily regardless of initial positions and orientations of both robot and target. The process to reach a target consisted of two phases, the orientation phase and acquiring phase. To reach a designated target, the robot

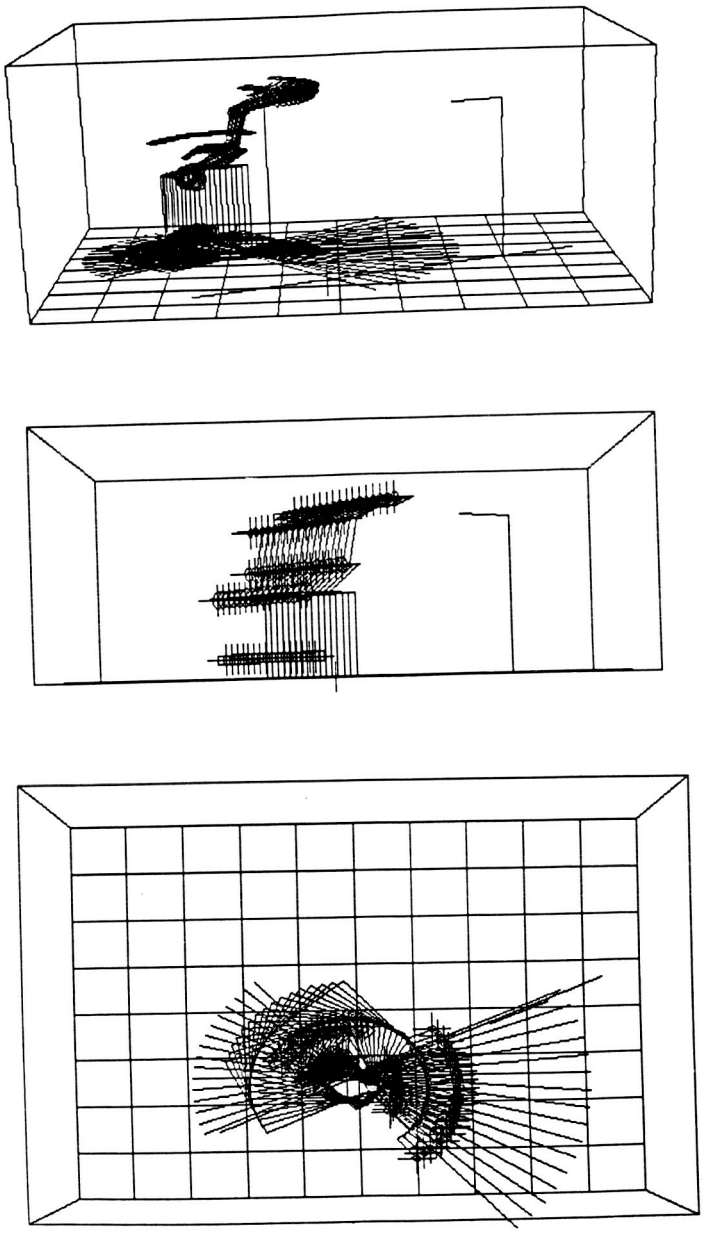


Fig. 5. Model approaching target: In the autonomous mode, upon receiving command to reach target, robot first performs orientation and then position relative to the target. Top: robot rotates to safe zone location. Middle: Then moves forward to this point. Bottom: Rotates to align with target.

first rotated (Fig. 5, upper view) until its new direction intersected with target direction at the safe zone location (Fig. 5; Fig. 3, right panel --- large plus signs superimposed on target approach line). Next the robot moved forward until it reached this location (Fig. 5, middle view). It then performed a second rotation so that its direction aligned with the target direction (Fig. 5, lower view). Thus the robot completed the first orientation phase in approaching the target. Since accuracy was not crucial in this phase, visual update was more relaxed and faster control speed could be obtained. Once the direction of the robot was in alignment with its target, the second phase began. The image processing computer immediately switched to a fast operating mode, closing the feedback loop. The 3D knowledge model carefully cruised the robot and guided the robot gripper to finally acquire the target (Fig. 2,4,5).

DISCUSSION

Since all tasks have been performed by the AT-386, our programs have grown close to the limit of the MS-DOS capability. Compromises have had to be made among competitive issues such as performance, memory utilization and implementation of new schemes. To alleviate this problem, the 3D knowledge-based model will be ported over to the SUN-386 workstation acting as the local control station (Fig. 1). It will oversee the image processing tasks and the control of the robots that will remain with the AT 386 computer in the remote station. The Iris graphics workstation will provide the display to the human operator at the earth laboratory.

Future research will include systematic benchmnark studies for the various image processing schemes as they fail while becoming subject to extreme conditions. Controllable cameras and wider working environments for the robots will also be utilized.

In conclusion, the top-down approach [20] with 3D model control plays a crucial role in resolving many conflicting image processing problems that are inherent in the bottom-up approach of most current machine vision processes. The 3D model control approach is also capable of providing the necessary visual feedback information for both the control algorithms and for the human operator. Finally, it provides an extreme reduction in communication, the mostly needed feature in telerobotics applications.

ACKNOWLEDGEMENTS: - We wish to thank Dr. A. K. Bejczy and Dr. Stephen Ellis, the technical monitors of our JPL and NASA/Ames research grants, and as well our colleagues in the Telerobotics Unit, UCB.

REFERENCES

1. Alt, F.L., "Digital Pattern Recognition by Moments", JACM 9:240-258 (1962).
2. Butler, J.W., Butler, M.K., and Stroud, A., "Automatic Classification of Chromosomes," K. Enslein, ed., Data Acquisition and Processing in Biology and Medicine, 3, Pergamon, New York (1964)
3. Canny, John, "A Computational Approach to Edge Detection", IEEE Trans. on Pattern Anal. and Machine Intel. PAMI-8 (1986).
4. Castleman, K. R., Digital Image Processing, Prentice Hall (1979).
5. Ferrell, W.R., Sheridan, T.B., "Supervisory Control of Remote Manipulation", IEEE Spectrum 4: 81-88 (1967).
6. Hall, E.L., Computer Image Processing & Recognition, Academic, N. Y. (1979).

7. Hirose, M., Sudo, J., and Ishii, T., "Human Interface of Remote Operation in Three-Dimensional Space," 2nd Symposium on Human Interface. Tokyo (1986).
8. Horn, B. K. P., "Robot Vision," MIT Press and McGraw-Hill (1986).
9. Hu, M.K., "Visual Pattern Recognition by Moment Invariants," IRE Trans. Inf. Theory, IT-8: 179-187 (1962).
10. Kim, W.S., Ellis, S., Tyler, M., Hannaford, B., and Stark, L., "Quantitatively Evaluation of Perspective and Stereoscopic Displays in Three-Axis Manual Tracking Tasks," IEEE System, Man and Cybernetics 16: 61-72 (1987).
11. Kim, W.S., Tendick, F., and Stark, L., "Visual Enhancement in Pick-and-Place Tasks: Human Operators Controlling a Simulated Cylindrical Manipulator", IEEE J. Rob. and Auto RA-3: 418-425 (1987).
12. Marr, D., Vision Freeman, San Francisco (1982).
13. Nguyen, A.H., Ngo, H., and Stark L., "Robotic Model Control of Image Processing", Proc. IEEE Intl. Conf. Sys., Man & Cyb., Beijing (1988)
14. Pratt, W. K. Digital Image Processing, Wiley (1978). <Y7
15. Reutz, P. and Brodersen, R., "An Image Recognition System Using Algorithmically Dedicated Integrated Circuits," Machine Vision Applic 1: 3-22 (1988).
16. Rosenfeld, A. and Kak, A. C., Digital Picture Processing, vols. 1 and 2, Academic (1982).
17. Sobel, I., "On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes," Art. Intel. 5: 185-198 (1974).
18. Stark, L., Kim, W., Tendick, F., et al., "Telerobotics: Display, Control, and Communication Problems," IEEE J Robotics Automation, RA-3: 67-75 (1987).
19. Noton, D., Stark, L. "Eye Movements and Visual Perception," Scientific American 224: 34-43 (1969).
20. Stark, L, Mills, B., Nguyen, A., and Ngo, H, "Instrumentation and Robotic Image Processing Using Top-Down Model Control", Robotics and Manufacturing, Jamshidi et al, eds., ASME, NY, 675-682 (1988).

WEIGHTED FEATURE SELECTION CRITERIA FOR VISUAL SERVOING OF A TELEROBOT

John T. Feddema, C. S. G. Lee, and O. R. Mitchell

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

Because of the continually changing environment of a space station, visual feedback is a vital element of a telerobotic system. A real-time visual servoing system would allow a telerobot to track and manipulate randomly moving objects. This paper develops methodologies for the automatic selection of image features to be used to visually control the relative position between an eye-in-hand telerobot and a known object. A weighted criteria function with both image recognition and control components is used to select the combination of image features which provides the best control. Simulation and experimental results of a PUMA robot arm visually tracking a randomly moving carburetor gasket with a visual update time of 70 milliseconds are discussed.

I. Introduction

Most would agree that the eventual goal of a telerobot is to perform dangerous tasks in space which would otherwise require human intervention. To perform these tasks, telerobots must be equipped with many of the sensory capabilities of humans. Because of the continually changing environment of a space station, vision is undoubtably a very important sense. Until recently, the primary uses of vision in telerobotics have been for recognizing, locating, and inspecting stationary parts. Image processing equipment is now reaching the stage where vision may be used as a feedback signal to control the position and orientation (pose) of the telerobot's end-effector in real time [1][2]. This visual feedback would allow a telerobot to manipulate and track a randomly moving part without any previous knowledge of the part's placement or motion.

The type of feedback for visual servoing systems has taken two forms [3]: position-based and feature-based. The traditional method has been to extract image features †, recognize the desired object by matching image features to a set of pre-taught features, interpret the pose of the object based on the image features, and use the error between desired and estimated pose to drive the system. A second method is to use actual image features instead of the part's interpreted position as the feedback signal for controlling the manipulator [4]. With the proper selection, these features can be directly related to the control parameters of the robotic system. The savings in time needed to interpret the workpiece's pose from the image features is made possible by determining the desired image features during an off-line CAD simulation.

In this paper, a resolved motion rate control scheme [5] with feature-based feedback are used to visually servo a robot manipulator with an eye-in-hand camera over a moving object (see Figure 1). This method of visual feedback introduces two fundamental questions. How many image features are necessary to control the desired degrees of freedom of the manipulator end-effector? And which image features would provide the best control? This paper addresses these two questions and develops methodologies for the automatic selection of image features used to visually control the relative pose between the manipulator end-effector and a workpiece. The selection of these features depends on a blend of image recognition and control criteria. The image recognition criteria include feature robustness, completeness, uniqueness, and cost of feature extraction. The control criteria include system observability, controllability, and sensitivity. A weighted criteria function is used to select the combination of image features which provides the best control. Both computer simulations and laboratory experiments on a PUMA robot arm were conducted to verify the performance of the feature selection criteria.

This work was supported in part by an IBM research fellowship and in part by the National Science Foundation under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

†In this paper, image features refer to geometric shapes in the camera's image. Examples include circles, edges, corners, and curves.

II. Differential Relationship between Part's Pose and Image Feature Points

In the resolved motion rate control structure in Figure 1, the changes in image features are transformed into changes in joint angles. This transformation may be decomposed into a series of three transformations: feature to camera coordinates, camera to end-effector coordinates, and end-effector to robot joint coordinates. The last two transformations are well known [6] but the first is not. For our purposes, we will assume that a unique transformation from camera space to robot joint space exists. Therefore, our ability to control the pose of the robot depends on the differential transformation from image feature space to camera space. This section concentrates on the differential transformation from image feature points †† to camera coordinates.

When analyzing the transformations from the camera space to the image feature space, consider the coordinate frames shown in Figure 2. The following nomenclature is used:

(x, y, z) = position of the part with respect to the camera frame;

(ϕ, θ, ψ) = roll, pitch, and yaw orientation of the part with respect to the camera frame;

$({}^{p}x_i, {}^{p}y_i, {}^{p}p_i)$ = position of feature point i on the part with respect to the part frame;

$({}^{c}x_i, {}^{c}y_i, {}^{c}z_i)$ = position of feature point i on the part with respect to the camera frame;

$({}^{I}x_i, {}^{I}y_i)$ = corresponding position of the point in the image plane;

f = focal length of the camera lens;

γ_x = x axis scaling factor in pixels/mm due to camera sampling;

γ_y = y axis scaling factor in pixels/mm due to camera sampling; and

(x_o, y_o) = the image plane offset in pixels due to camera sampling.

We assume that the camera characteristics ($f, \gamma_x, \gamma_y, x_o, y_o$) are known. To be able to interpret the 3-D pose of a part, we have assumed that the spatial relationships between feature points, i.e., $({}^{p}x_i, {}^{p}y_i, {}^{p}z_i)$, are known from a CAD model.

Geometric optics are used to model the mapping between the Cartesian space and the image feature space. The mapping consists of two stages: a thin lens model of the perspective transformation, and a mapping into a two-dimensional plane caused by camera sampling. The transformation from the camera frame, c, to the image plane, I, can be written as

$$\begin{bmatrix} {}^{I}x_i w_i \\ {}^{I}y_i w_i \\ w_i \end{bmatrix} = \begin{bmatrix} \gamma_x & 0 & 0 & x_o \\ 0 & \gamma_y & 0 & y_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & f \\ 0 & 0 & -1 & f \end{bmatrix} \begin{bmatrix} {}^{c}x_i \\ {}^{c}y_i \\ {}^{c}z_i \\ 1 \end{bmatrix} \tag{1}$$

or

$${}^{I}x_i = \gamma_x \left[\frac{f}{f - {}^{c}z_i} \right] {}^{c}x_i + x_o \qquad \text{and} \qquad {}^{I}y_i = \gamma_y \left[\frac{f}{f - {}^{c}z_i} \right] {}^{c}y_i + y_o \tag{2}$$

where we assume that the blurring effects, the quantization, and the lens distortion effects are negligible.

Using the differential transform technique [6], the change in image positions $(d^{I}x_i, d^{I}y_i)$ in terms of the change in the part's pose $(d_x, d_y, d_z, \delta_x, \delta_y, \delta_z)$ is [7]

$$\begin{bmatrix} d^{I}x_i \\ d^{I}y_i \end{bmatrix} = \begin{bmatrix} J_{x_i d_x} & J_{x_i d_y} & J_{x_i d_z} & J_{x_i \delta_x} & J_{x_i \delta_y} & J_{x_i \delta_z} \\ J_{y_i d_x} & J_{y_i d_y} & J_{y_i d_z} & J_{y_i \delta_x} & J_{y_i \delta_y} & J_{y_i \delta_z} \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \tag{3}$$

where

$$J_{x_i d_x} = \frac{\gamma_x f}{f - {}^{c}z_i}, \qquad J_{x_i d_y} = 0, \qquad J_{x_i d_z} = \frac{\gamma_x f \, {}^{c}x_i}{(f - {}^{c}z_i)^2},$$

†† The (x, y) position of a feature such as a circle or corner in the image will be referred to as a feature point.

$$J_{x,\delta_x} = \frac{\gamma_x f\, {}^c x_i\, {}^c y_i}{(f - {}^c z_i)^2}, \qquad J_{x,\delta_y} = \frac{\gamma_x f}{f - {}^c z_i}\left[{}^c z_i - \frac{{}^c x_i^2}{f - {}^c z_i}\right], \qquad J_{x,\delta_z} = \frac{-\gamma_x f\, {}^c y_i}{f - {}^c z_i},$$

$$J_{y,\delta_x} = 0, \qquad J_{y,\delta_y} = \frac{\gamma_y f}{f - {}^c z_i}, \qquad J_{y,\delta_z} = \frac{\gamma_y f\, {}^c y_i}{(f - {}^c z_i)^2},$$

$$J_{y,\delta_x} = \frac{\gamma_y f}{f - {}^c z_i}\left[- {}^c z_i + \frac{{}^c y_i^2}{f - {}^c z_i}\right], \qquad J_{y,\delta_y} = \frac{-\gamma_y f\, {}^c x_i\, {}^c y_i}{(f - {}^c z_i)^2}, \quad \text{and} \quad J_{y,\delta_z} = \frac{\gamma_y f\, {}^c x_i}{f - {}^c z_i}.$$

This expression is suitable for simulation purposes where we want to determine the change in image features given a small change in the part's pose. However, for control purposes we seek an inverse solution. We would like to know the change in the part's pose given the change in several feature points in the image. An exact inverse may exist if three feature points are considered. The linear differential relationship between the change in image feature points and the change in the part's pose would then be 6×6 matrix **J**.

Notice that this Jacobian matrix depends on the positions of the feature points with respect to the camera. For the inverse Jacobian matrix **J** to exist, all three points can not have the same x and y position in the image plane. If the points are collinear, then a rotation about the line would not be observable. Another restriction is that not all feature points can be on a plane perpendicular to the focal axis with one of the points located at the focal center. Again, there exists a rotation in this plane which the camera will not be able to sense.

One final problem still remains. How can we use this transformation if we do not know the actual positions of the feature points with respect to the camera? In our experiments, two approaches were used. The first was to estimate their positions from their image positions and the known spatial relationships between image points. The second was to use the desired positions as determined by the CAD simulation. While the latter is self-explanatory, the first method deserves further development.

The objective is to find the pose of the part with respect to the camera, i.e., $(x, y, z, \phi, \theta, \psi)$, from the a priori information, $({}^p x_i, {}^p y_i, {}^p z_i)$, and the measured feature points in the image, $({}^I x_i, {}^I y_i)$, for $i = 1, 2, \cdots, n$, where n is the number of feature points in the image. Variations of this "location determination problem" have appeared in several papers [8]. Overdetermined systems of equations are typically used to determine the "best" solution. Least squares techniques, the Random Sample Consensus paradigm [8], and a generalized Hough transform approach [9] have been proposed to eliminate errors caused by noisy images and modeling errors.

Unfortunately, many of these methods are too time consuming for real-time use. Instead, we propose using the following gradient search to continually update the object's pose. This search minimizes the sum of squared distance errors between four or more actual feature points and the geometrically modeled feature points in the image. Because of the nature of this search, a fairly good initial estimate of the part's pose is necessary. It should also be noted that the resulting least squares solution is optimal if the image noise has a Gaussian distribution. However, if there are outliers in the data, the Random Sample Consensus approach [8] might provide more accurate results.

The objective of this gradient search is to vary the pose of the workpiece until the image positions of the modeled feature points align with the actual image feature points. In mathematical terms, we would like to minimize

$$F = \sum_{i=1}^{n} F_i = \frac{1}{2} \sum_{i=1}^{n} ({}^I x_i - u_i)^2 + ({}^I y_i - v_i)^2 \tag{4}$$

where (u_i, v_i) is the actual image position of feature point i. The modeled position of feature point i in the image is determined from the camera's perspective and sampling Eq. (2), and from the roll, pitch, yaw representation of the part with respect to the camera [6],

$$\begin{bmatrix} {}^c x_i \\ {}^c y_i \\ {}^c z_i \\ 1 \end{bmatrix} = \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi & x \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi & y \\ -s\theta & c\theta s\psi & c\theta c\psi & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^p x_i \\ {}^p y_i \\ {}^p z_i \\ 1 \end{bmatrix}, \tag{5}$$

where $c\phi \equiv cos(\phi)$, $s\phi \equiv sin(\phi)$, $c\theta \equiv cos(\theta)$, $s\theta \equiv sin(\theta)$, $c\psi \equiv cos(\psi)$, and $s\psi \equiv sin(\psi)$.

In these equations, the known variables are the camera parameters, $(f, \gamma_x, \gamma_y, x_o, y_o)$; the actual image feature positions, (u_i, v_i), for $i = 1, 2, \cdots, n$; and the positions of the features with respect to the part's frame, $({}^p x_i, {}^p y_i, {}^p z_i)$, for $i = 1, 2, \cdots, n$, where n is the number of features. The unknown variables (the ones we want to solve for) are the pose of the part, $(x, y, z, \phi, \theta, \psi)$. For ease of notation, let $\mathbf{x} = [x, y, z, \phi, \theta, \psi]^T$, $\mathbf{y}_i = [{}^c x_i, {}^c y_i, {}^c z_i]^T$, and $\mathbf{z}_i = [{}^I x_i, {}^I y_i]^T$, where the superscript T denotes vector/matrix transpose. Then Eq. (2) can be represented by $\mathbf{z}_i = \mathbf{G}(\mathbf{y}_i)$, and Eq. (5) can be represented by $\mathbf{y}_i = \mathbf{H}_i(\mathbf{x})$.

Newton's gradient search is used to minimize this error function with respect to the x parameters. The kth iteration of the search is given by

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \mathbf{F}_{\mathbf{xx}}^{-1}(\mathbf{x}_k)\, \nabla F^T(\mathbf{x}_k) \tag{6}$$

where ∇F is the gradient of F with respect to $\mathbf{x}$ and the $\mathbf{F}_{\mathbf{xx}}$ is the Hessian matrix of F with respect to $\mathbf{x}$.

Because the relationship between the error function and the part's pose consists of a series of composite mappings described in Eqs. (2), (4), and (5), the gradient of F is the product of the Jacobian matrices of individual mappings. The gradient of F with respect to $\mathbf{x}$ is

$$\nabla F = \sum_{i=1}^{n} \mathbf{F}_{iz_i}\, \mathbf{G}_{\mathbf{y}_i}\, \mathbf{H}_{i\mathbf{x}} \tag{7}$$

where $\mathbf{F}_{iz_i}$ is the Jacobian matrix of F_i with respect to $\mathbf{z}_i$, $\mathbf{G}_{\mathbf{y}_i}$ is the Jacobian matrix of $\mathbf{G}$ with respect to $\mathbf{y}_i$, and $\mathbf{H}_{i\mathbf{x}}$ is the Jacobian matrix of $\mathbf{H}_i$ with respect to $\mathbf{x}$. The elements of these Jacobian matrices are given in [7].

The chain rule for the Hessian matrix $\mathbf{F}_{\mathbf{xx}}$ is slightly more difficult.

$$\mathbf{F}_{\mathbf{xx}} = \sum_{i=1}^{n} \mathbf{H}_{i\mathbf{x}}^T\, \mathbf{G}_{\mathbf{y}_i}^T\, \mathbf{F}_{iz_iz_i}\, \mathbf{G}_{\mathbf{y}_i}\, \mathbf{H}_{i\mathbf{x}} + \frac{\partial F_i}{\partial {}^I x_i} \frac{\partial^2 ({}^I x_i)}{\partial \mathbf{x}^2} + \frac{\partial F_i}{\partial {}^I y_i} \frac{\partial^2 ({}^I y_i)}{\partial \mathbf{x}^2} \tag{8}$$

where the Hessian matrix of ${}^I x_i$ with respect to $\mathbf{x}$ is

$$\frac{\partial^2 ({}^I x_i)}{\partial \mathbf{x}^2} = \mathbf{H}_{i\mathbf{x}}^T \frac{\partial^2 ({}^I x_i)}{\partial \mathbf{y}_i^2} \mathbf{H}_{i\mathbf{x}} + \frac{\partial {}^I x_i}{\partial {}^c x_i} \frac{\partial^2 ({}^c x_i)}{\partial \mathbf{x}^2} + \frac{\partial {}^I x_i}{\partial {}^c y_i} \frac{\partial^2 ({}^c y_i)}{\partial \mathbf{x}^2} + \frac{\partial {}^I x_i}{\partial {}^c z_i} \frac{\partial^2 ({}^c z_i)}{\partial \mathbf{x}^2}, \tag{9}$$

and the Hessian matrix of ${}^I y_i$ with respect to $\mathbf{x}$ is

$$\frac{\partial^2 ({}^I y_i)}{\partial \mathbf{x}^2} = \mathbf{H}_{i\mathbf{x}}^T \frac{\partial^2 ({}^I y_i)}{\partial \mathbf{y}_i^2} \mathbf{H}_{i\mathbf{x}} + \frac{\partial {}^I y_i}{\partial {}^c x_i} \frac{\partial^2 ({}^c x_i)}{\partial \mathbf{x}^2} + \frac{\partial {}^I y_i}{\partial {}^c y_i} \frac{\partial^2 ({}^c y_i)}{\partial \mathbf{x}^2} + \frac{\partial {}^I y_i}{\partial {}^c z_i} \frac{\partial^2 ({}^c z_i)}{\partial \mathbf{x}^2}. \tag{10}$$

The elements of these Hessian matrices are also in [7].

III. Feature Selection Criteria

Since the Jacobian obtained in the previous section depends on the features' positions in the 3-D space with respect to the camera, some features will provide better visual control than others as the part moves with respect to the camera. In addition to these control issues, image recognition plays an important role in choosing features which are reliable and robust. This section lists several image recognition and control criteria which should be considered in the selection process. This is not an all inclusive list and additional criteria could be added at a latter date.

The image recognition criteria used in our feature selection process include [10]:

1. rare features (similar to unique features in [11]),
2. feature set robustness (similar to likelihood of being seen in [11]),
3. computational inexpensive features, and
4. feature set completeness.

To quantitatively evaluate these criteria, a set of measures was designed for the visual servoing experiments. For uniformity, these measures were designed to range between zero and one with zero being the most desirable and one being the least desirable.

A unique feature has an easily identifiable characteristic which differs from other features in the image. Such a feature is often used to quickly identify an object. This makes the feature very useful for control since the feature can be quickly re-identified if it is momentarily lost. A measure of feature uniqueness for a set of features $\{f_1, \cdots, f_m\}$ may be written as

$$\chi_1(f_1, \cdots, f_m) = \frac{1}{Nm(n-1)} \left[\sum_{i=1}^{m} \sum_{j=1}^{N} M_{ij} \right] \tag{11}$$

where m is the number of features to be used for control, n is the total number of features in the images, N is the number of possible views during the control process, and M_{ij} is the number of similar features to feature i in image j ($0 \le M_{ij} \le n-1$). Note that the measure χ_1 reaches a maximum of one if $M_{ij} = n-1$ for all i and j and a minimum of zero if $M_{ij} = 0$ for all i and j.

In order that the recognition process be resilient to noise, we would like the features to be robust. For feature extraction systems where the image is scanned for features, feature robustness depends on the size and type of feature and the method of feature extraction. Usually the larger the feature is in the image, the more robust the feature is. A general measure of feature robustness may be written as:

$$\chi_2(f_1, \cdots, f_m) = \frac{1}{Nm}\left[\sum_{i=1}^{m}\sum_{j=1}^{N}(1-\sigma_{ij})\,\rho_{ij}\right] \tag{12}$$

where $0 \le \sigma_{ij} \le 1$ is proportional to the size of the feature i in the image j, and $0 \le \rho_{ij} \le 1$ is related to the reliability of feature extraction method ij. In our experiments, features were restricted to circles in the image. Since the same method of feature extraction was used throughout the experiments, the reliability factor is $\rho_{ij} = 1$ for all i,j. Our measure of feature robustness for circles was

$$\chi_2(f_1, \cdots, f_m) = 1 - \frac{1}{Nmr_{\max}}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} r_{ij}\right] \tag{12.a}$$

where r_{ij} was the radius of circle j in image j, and $r_{\max}$ was the maximum radius possible.

The computational expense of features refers to the time and space complexities of the feature extraction process. For most cases, space complexity is negligible. Time complexity, on the other hand, is very important for determining the tracking ability of the visual servoing control. The shorter the time of feature extraction is, the larger the bandwidth of the control. For feature extraction systems where the image is scanned for features, computational expense also depends on the size and type of feature and the method of feature extraction. In contrast to the robustness criteria, usually the smaller the feature is in the image, the smaller the computational expense is. A general measure of computational expense of features may be written as:

$$\chi_3(f_1, \cdots, f_m) = \frac{1}{Nm}\left[\sum_{i=1}^{m}\sum_{j=1}^{N}\sigma_{ij}\,\tau_{ij}\right] \tag{13}$$

where $0 \le \tau_{ij} \le 1$ is related to the time complexity of the feature extraction method ij for a feature of fixed size. In our experiments, the location of a circle was verified with four scan lines spanning out from the approximate origin of the circle to the circle's edge. Again, since the same method of feature extraction was used throughout the experiments, $\tau_{ij} = 1$ for all i,j. Our measure of computational expense for the circles was

$$\chi_3(f_1, \cdots, f_m) = \frac{1}{Nmr_{\max}}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} r_{ij}\right]. \tag{13.a}$$

Notice the trade-off between the computational expense and robustness criteria. Smaller circles have less computational cost with poor feature robustness, while larger circles have better robustness at the expense of computational cost.

If a workpiece can be identified from any view point, the set of features is said to be complete. For our circumstances where we are continually controlling the position of the workpiece with respect to the camera, only a subset of all possible views is necessary. Our measure for feature set completeness is

$$\chi_4(f_1, \cdots, f_m) = \frac{1}{Nm}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} u_{ij}\right] \qquad \textit{where} \qquad u_{ij} = \begin{cases} 0, & \text{if feature } i \text{ is in image } j \\ 1, & \text{if feature } i \text{ is not in image } j. \end{cases} \tag{14}$$

The control criteria used in our feature selection process include:

1. Observability of the workpiece's pose through image feature points.
2. Controllability of the workpiece's pose using the inverse Jacobian matrix in section II.
3. Sensitivity of the control to noise.

In the presense of image noise, the gradient search in section II was used to minimize the error in the image positions of four or more features. Since this error is measurable, it provides a means of evaluating the observability of the part's pose with respect to the camera. Using the error function F in Eq. (22), our measure for the observability of the part's pose is

$$\chi_5(f_1, \cdots, f_m) = \frac{1}{NF_{\max}}\sum_{j=1}^{N}\hat{F} \qquad \textit{where} \qquad \hat{F} = \begin{cases} F, & \text{if } F \le F_{\max} \\ F_{\max}, & \text{if } F > F_{\max}, \end{cases} \tag{15}$$

and $F_{\max}$ is the largest acceptable error.

If the sampling time between image acquisitions is short and the distance between the actual and desired image features is small, the differential relationship in section II could be used to control the relative pose of the workpiece. The desired change in camera position would be determined by multiplying the difference between actual and desired features by the inverse Jacobian. In this respect, we will say that the pose of the workpiece is "controllable" if the inverse Jacobian exists and is non-singular.

In addition to checking that a unique solution exists, we would like the equations to be well-conditioned. This means that for "small" changes in **J** and the part's pose, the changes in image feature positions should also be small. This could be thought of a measure of the control's sensitivity to noise. The condition of the Jacobian matrix **J** is

$$c(\mathbf{J}) = \|\mathbf{J}\| \, \|\mathbf{J}^{-1}\| \tag{16}$$

where the norm may be $\|\cdot\|_1$, $\|\cdot\|_2$, or $\|\cdot\|_\infty$. Moderately small values of $c(\mathbf{J})$ imply that the equations are well-conditioned. However, large values of $c(\mathbf{J})$ do not necessarily mean that the equations are ill-conditioned. Instead, it just means that the equations will be ill-conditioned for some changes in the part's pose. In general, we would like to choose image feature points which would minimize the condition of **J**. Since the condition of **J** will become extremely large as the Jacobian approaches a singular point, the condition may also be used to evaluate the controllability of the workpiece's pose using the inverse Jacobian matrix. Therefore, a measure for evaluating both the controllability and sensitivity of the system may be written as

$$\chi_6(f_1, f_2, f_3) = \frac{1}{N c_{\max}} \sum_{j=1}^{N} \hat{c}(\mathbf{J}) \qquad \textit{where} \qquad \hat{c}(\mathbf{J}) = \begin{cases} c(\mathbf{J}), & \text{if } \mathbf{J}^{-1} \text{ exists and } c(\mathbf{J}) \le c_{\max}, \\ c_{\max}, & \text{if } \mathbf{J}^{-1} \text{ exists and } c(\mathbf{J}) > c_{\max}, \\ c_{\max}, & \text{if } \mathbf{J}^{-1} \text{ does not exist}, \end{cases} \tag{17}$$

and $c_{\max}$ is the largest acceptable value for the condition of **J**.

Another consideration for real-time control is the effect that changes in the feature positions have on the elements of the Jacobian matrix. To reduce computational costs, it is desirable for the elements of the Jacobian to be constant or slowly time-varying. If the elements do not change substantially between two camera acquisitions, the inverse Jacobian does not have to be updated each sample time. Therefore, we would like to choose image feature points such that the elements of the Jacobian change very little throughout the motion. In other words, we would like to minimize the sensitivity of **J** to the change of image feature points,

$$s(\mathbf{J}) = \sum_{i=1}^{3} \sum_{k=0}^{6} \sum_{m=0}^{6} \left\{ \left| \frac{\partial \mathbf{J}_{km}}{\partial {}^{c}x_i} \cdot \frac{{}^{c}x_i}{\mathbf{J}_{km}} \right| + \left| \frac{\partial \mathbf{J}_{km}}{\partial {}^{c}y_i} \cdot \frac{{}^{c}y_i}{\mathbf{J}_{km}} \right| + \left| \frac{\partial \mathbf{J}_{km}}{\partial {}^{c}z_i} \cdot \frac{{}^{c}z_i}{\mathbf{J}_{km}} \right| \right\} \tag{18}$$

where $\mathbf{J}_{km}$ are the elements of the Jacobian. Our measure of sensitivity to change was

$$\chi_7(f_1, f_2, f_3) = \frac{1}{N s_{\max}} \sum_{j=1}^{N} \hat{s}(\mathbf{J}) \qquad \textit{where} \qquad \hat{s}(\mathbf{J}) = \begin{cases} s(\mathbf{J}), & \text{if } s(\mathbf{J}) \le s_{\max}, \\ s_{\max}, & \text{if } s(\mathbf{J}) > s_{\max}, \end{cases} \tag{19}$$

and $s_{\max}$ is the largest acceptable value of $s(\mathbf{J})$.

Because of the various conflicting interests of the above measures, a weighted criteria function was used to select the features which would provide the best control

$$\chi(f_1, \cdots, f_m) = \sum_{i=1}^{7} w_i \, \chi_i(f_1, \cdots, f_m) \tag{20}$$

where $\sum_{i=1}^{7} w_i = 1$ are the weighting factors which are the choice of the designer. Similar to the individual measures, the best selection will be the set of features with the smallest overall measure χ.

IV. Simulation and Experimental Results

Both computer simulations and laboratory experiments were performed to illustrate the performance of the proposed feature selection criteria. Computer simulations were used to test the performance of the feature selection criteria under ideal conditions and with additive image noise. The experiments verified the simulation results and showed the usefulness of the feature selection criteria for visually controlling a robot manipulator in real time.

In our experiments, a single Pulnix TM-540 CCD camera mounted on the end-effector of a PUMA 600 robot was used to visually servo the robot's end-effector over a moving carburetor gasket (the workpiece) as shown in Figure 3. Because of their ease of feature extraction, the circles in the gasket were used as the control features. The

visual servo control was initiated when the camera was approximately over the gasket in a pre-taught position. After the gasket was recognized, the locations of three circles in the image were continually updated. The changes in camera pose needed to control the robot were determined by multiplying the difference between the desired and actual image positions by the inverse Jacobian in section II. The changes in camera pose were then converted to the changes in joint angles using the manipulator's inverse Jacobian [12].

The simulations were modeled after the experiments and their objective was to determine which three image feature points (circles) of the carburetor gasket out of a total of seven circles (see Figure 3) would be best suited for controlling the pose of the gasket with respect to the camera. Table 1 lists the positions with respect to the gasket's frame and the radii of the seven largest holes. The number of feature points was limited to seven to reduce the number of possible combinations. The desired position and orientation of the gasket with respect to the camera was (x, y, z) = (-48,63,200) millimeters and (roll, pitch, yaw) = (-90,0,0) degrees, respectively. The camera parameters are given in Table 2.

First, we evaluated the condition and sensitivity of J for all possible combinations of three circles. We found that the set of features which minimize $c(\mathbf{J})$ form an equilateral triangle about the origin of the gasket. In particular, the set of circles {0,3,5} minimized $c(\mathbf{J})$ while the set of circles {0,1,2} maximized $c(\mathbf{J})$. We also found that the condition of J decreases as the gasket is moved closer to the camera. On the other hand, the sensitivity of J to change was minimized as the feature points moved closer to the focal axis and the x and y image axes. According to this sensitivity criteria, the best set of features would be the set of circles {1,2,4}. In contrast to the condition of J, the sensitivity of J to change is minimized if the gasket is as far away from the camera as possible. If computation time was a factor, a weighted sum of the measures χ_6 and χ_7 could be used to determine a set of features which would provide accurate control without updating the inverse Jacobian.

Next, we considered the tracking response of the camera for the two sets of features, {0,3,5} and {0,1,2}. For both sets of features, circle 6 was used as a fourth point for determining location. To test the control process, a ramp input in position and orientation was applied to the gasket's pose. The change in the x, y, and z directions was 5 millimeters per sample. The change in the roll, pitch, and yaw of the gasket was 0.9 degrees per sample. If our system was operating at video rate (30 Hz - both fields), this would correspond to a positional velocity of 15 centimeters per second and a rotational velocity of 27 degrees per second. Several cases were run with and without image noise.

First consider the ideal case with no image noise. Table 3 shows the root mean square (RMS) error between the desired and actual position and orientation of the workpiece with respect to the camera. Three schemes were used to update the Jacobian matrix. In column 1, the Jacobian was updated with the actual positions of the feature points. In column 2, the Jacobian was updated only once with the desired positions of the feature points. In column 3, the Jacobian was updated with the estimated positions of the feature points using the methods in section II. The following conclusions were made from the ideal case:

1. Updating the Jacobian with the actual positions provides the best control.
2. When the actual or desired positions of the features were used, the set of features {0,3,5} performed as well as the feature set {0,1,2}.
3. As predicted by the large condition of the Jacobian for feature set {0,1,2}, it was difficult to estimate the part's pose from the feature set {0,1,2}. Therefore, for estimation purposes it is best to choose a set of feature points with a small condition number, such as the feature set {0,3,5}.

Next consider the same simulation except add Gaussian image noise with a standard deviation of 0.5 pixels. The RMS error for this case is shown in Table 4. In addition to the same columns as in Table 3, a fourth column shows the results when seven points are used to estimate the pose of the workpiece. The columns with ∞ indicate that the workpiece was lost before the simulation was completed. The following conclusions can be made from the noisy image case:

1. Updating the Jacobian with the desired positions provides the best control.
2. Even when using desired or actual positions to update the Jacobian, image noise has a costly effect on the resolved motion rate control of a set of features with a small condition number, such as the feature set {0,1,2}. Therefore, for control purposes it is best to choose a set of feature points with a small condition number, such as the feature set {0,3,5}.
3. The more points used to estimate the part's pose, the better the estimate will be.

In the experiments, the camera was first moved to the desired position over the gasket and the selection criteria were evaluated as shown in Table 5. The criteria weighting factors were set to $\{w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$ = {0.1, 0.15, 0.05, 0.1, 0.2, 0.2, 0.2}. The computation time of feature extraction was weighted less than the other criteria since there was not a noticeable difference in delay time when extracting one circle over the next. The maximum acceptable pose estimation error, condition, and sensitivity were F_{max} = 100, c_{max} = 100,000, and s_{max} = 68.

After the features were selected, the gasket was placed on a conveyor belt moving at a speed of 2.7 centimeters per second in the y and z directions of the world coordinates. Using the resolved motion rate visual feedback and a feature-based trajectory generator [12], the PUMA robot tracked the gasket. The feature-based trajectory generator had a smoothing effect on image noise by spreading out the change in image features over time. To speed up real-time computation and avoid estimation noise effects the desired positions of the feature points were used to compute the Jacobian. This works satisfactorily if the feature points are already in the vicinity of the desired positions.

In Figure 4, the first 37 seconds show the ramp response of the system along the z direction while the last 10 seconds show the steady-state error when the conveyor belt was stopped. The response in the y direction was a similar ramp shaped graph. The responses in the x direction and the orientation angles were small oscillatory signals about the desired values. The maximum positional errors in the x, y, and z directions were 18.4, 21.6, and 8.0 millimeters, respectively. The maximum orientation errors in roll, pitch, and yaw were 1.5, 2.6, and 2.2 degrees, respectively. The oscillation in the x position and the orientation angles can be attributed to image noise and the feedback delay time. In order to increase the vision sampling, the verification process used to update the position of the circles was very simple. Unfortunately, it was also fairly noisy. During the servoing process, the circles in the image were located with an accuracy of $\pm$ 5 pixels. The delay time between when the image was taken and when the robot actuation began was approximately 100 milliseconds.

V. Conclusion

Methodologies for the automatic selection of image features used to visually control the relative pose between the manipulator end-effector and a workpiece were developed and analyzed. A resolved motion rate control scheme was used to update the robot's pose based on the position of three features in the camera's image. The selection of these three features for control was based on a weighted criteria function with both image recognition and control components. Because of the real-time nature of the control process, it is important to find reliable image features which can be quickly extracted from the image. Of particular importance for resolved motion rate control was the condition of the Jacobian matrix relating the differential change in the workpiece's pose to the corresponding differential change in the image feature points. To minimize the effects of image noise on the control, the condition of the Jacobian should be minimized. In the simulations and experiments, we found that estimating the pose of the workpiece was a time consuming and noise sensitive process. Because of this, the best tracking results were obtained by using the desired feature locations to compute the Jacobian instead of the estimated locations.

VI. References

[1] B. Wilcox, et al. "The Sensing and Perception Subsystem of NASA Research Telerobot," *Proceeding of the Workshop on Space Telerobotics*, G. Rodriguez, editor, JPL Publ. 87-13, July 1, 1987.

[2] P. Symosek, et al. "Knowledge-Based Vision for Space Station Object Motion Detection, Recognition, and Tracking," *Proceeding of the Workshop on Space Telerobotics*, JPL Publ. 87-13, July 1, 1987.

[3] A.C. Sanderson and L.E. Weiss, "Adaptive Visual Servo Control of Robots," Reprinted in *Robot Vision*, Alan Pugh, editor, IFS Publications Ltd., 1983.

[4] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic Sensor-Based Control of Robots with Visual Feedback," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 5, pp. 404-417, October 1987.

[5] D.E. Whitney, "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 122, pp. 303-309, December 1972.

[6] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, Cambridge, Massachusetts, 1981.

[7] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell, "Optimal Selection of Image Features for Resolved Rate Visual Feedback Control," Purdue University, Technical Report TR-ERC 89-3, January 1989.

[8] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. of the ACM*, Vol. 24, No. 6, pp. 381-395, June 1981.

[9] S. Linnainmaa, D. Harwood, and L.S. Davis, "Pose Determination of a Three-Dimensional Object Using Triangle Pairs," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 5, pp. 634-647, Sept. 1988.

[10] C. Hansen and T. Henderson, "CAGD-Based Computer Vision," *IEEE Workshop on Computer Vision*, pp. 100-105, November 1987. Ellis Horwood Publishers, New York, pp. 81-84, 1985.

[11] R. Horaud and T. Skordas, "Model-Based Strategy Planning for Recognizing Partially Occluded Parts," *Computer*, pp. 58-64, August 1987.

[12] J.T. Feddema and O.R. Mitchell, "Vision Guided Servoing with Feature-Based Trajectory Generation," *SME World Conference on Robotic Research*, Gaithersburg, Maryland, May 9-12, 1989.

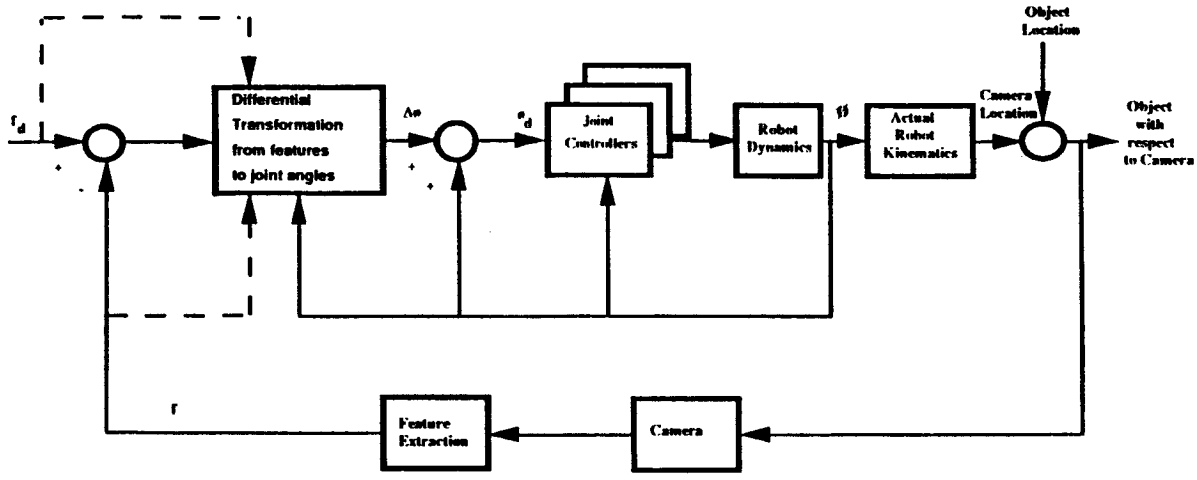


Figure 1. Resolved motion rate control structure for visual feedback.

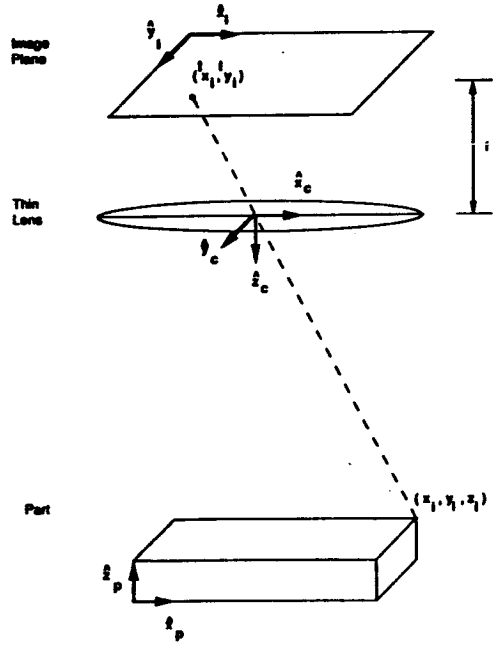


Figure 2. Part, camera, and image coordinate frames.

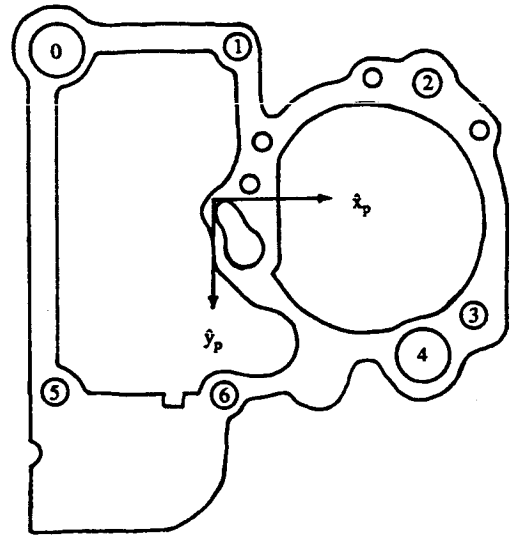


Figure 3. Seven image feature points of the carburetor gasket.

Table 1. Positions and radii of the circles with respect to the gasket's frame.

Positions and radii in millimeters				
No.	x	y	z	radius
0	98	5	0	5
1	98	42	0	3
2	93	80	0	3
3	46	91	0	3
4	37	81	0	5
5	28	6	0	3
6	28	39	0	3

Table 2. Intrinsic parameters of the camera.

Parameter	Symbol	Value
Focal length	f	8.0 mm
X scale factor	γ_x	-67.2832 pixels/mm
Y scale factor	γ_y	-84.7279 pixels/mm
X focal center	x_o	240.0 pixels
Y focal center	y_o	240.0 pixels

Table 3. Ideal case: RMS error of gasket with respect to the camera.

Control Features	Experiment		#1	#2	#3
	Jacobian Update		Actual	Desired	Estimated *
{0,3,5}	RMS Estimation Error	Position (mm)	0	0	0.000034
		Orientation (deg)	0	0	0.000032
	RMS Control Error	Position (mm)	0.2847	0.3526	0.2847
		Orientation (deg)	0.1493	0.1722	0.1493
{0,1,2}	RMS Estimation Error	Position (mm)	0	0	7.6440
		Orientation (deg)	0	0	4.5745
	RMS Control Error	Position (mm)	0.1987	0.1798	0.9727
		Orientation (deg)	0.0360	0.0430	0.2940

* Jacobian updated only if least squares search error F is small.

Table 4. Image noise case: RMS error of gasket with respect to the camera. The image noise had a Normal distribution with zero mean and a standard deviation of 0.5 pixels. The table shows the mean position and orientation RMS error of 10 trials.

Control Features	Experiment		#1	#2	#3	#4
	Jacobian Update		Actual	Desired	Estimate #1 *	Estimate #2 **
{0,3,5}	RMS Estimation Error	Position (mm)	0	0	2.5663	2.2569
		Orientation (deg)	0	0	2.7393	2.1801
	RMS Control Error	Position (mm)	1.2708	1.1542	1.5735	1.4578
		Orientation (deg)	1.3395	1.2593	1.5454	1.4490
{0,1,2}	RMS Estimation Error	Position (mm)	0	0	∞	∞
		Orientation (deg)	0	0	∞	∞
	RMS Control Error	Position (mm)	∞	∞	∞	∞
		Orientation (deg)	∞	∞	∞	∞

* Estimate #1 implies that four circles were used to estimate the gasket's pose.
** Estimate #2 implies that seven circles were used to estimate the gasket's pose.

Table 5. The best combinations of features with weights {0.1, 0.15, 0.05, 0.1, 0.2, 0.2, 0.2}.

Feature Selection Table For Experiments											
Circles				Selection Criteria							
#1	#2	#3	#4	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6	χ_7	χ
0	3	5	4	0.341	0.219	0.781	0.000	0.023	0.110	0.919	0.316
0	4	5	3	0.341	0.219	0.781	0.000	0.033	0.148	0.916	0.325
0	2	4	3	0.341	0.219	0.781	0.000	0.041	0.139	0.919	0.326
0	3	6	4	0.341	0.216	0.784	0.000	0.013	0.171	0.917	0.326
0	4	5	6	0.341	0.237	0.762	0.000	0.047	0.133	0.916	0.327
0	2	3	4	0.341	0.219	0.781	0.000	0.042	0.159	0.921	0.330
1	4	5	0	0.341	0.235	0.765	0.000	0.042	0.178	0.915	0.335
0	4	6	3	0.341	0.216	0.784	0.000	0.024	0.220	0.915	0.338
0	2	6	4	0.341	0.238	0.762	0.000	0.106	0.130	0.917	0.338
0	4	5	2	0.341	0.241	0.759	0.000	0.119	0.117	0.916	0.339

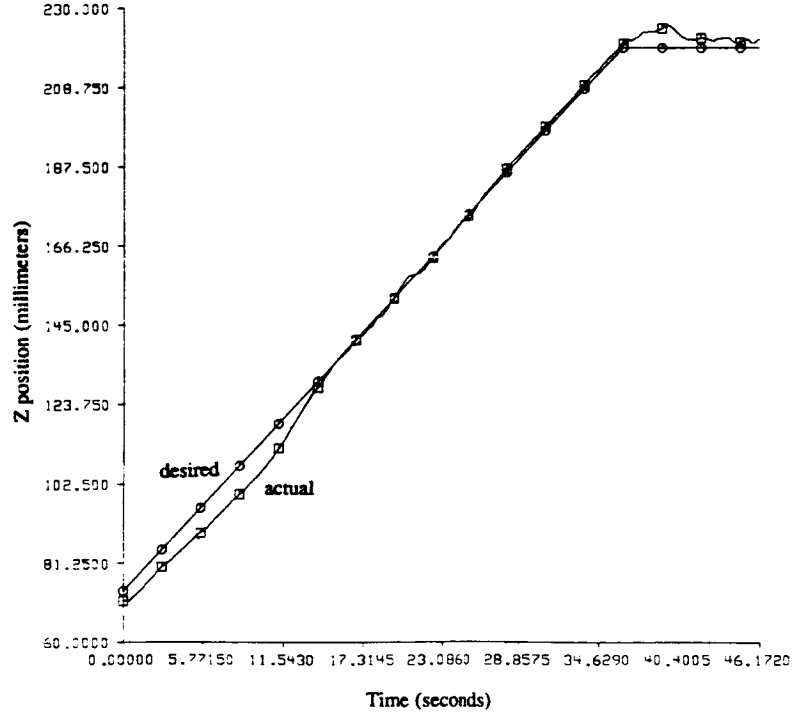


Figure 4. Robot end-effector's z position while tracking the gasket in the experiments.

Trinocular Stereovision using Figural Continuity, Dealing with Curved Objects

R. Vaillant and O.D. Faugeras
INRIA Domaine de Voluceau
BP 105, 78153 Le Chesnay Cedex France

Abstract

In this paper, we present a method to build a dense and reliable 3D description of a scene from three digital images by means of passive stereovision. Our method uses figural continuity to improve the results of a previously developed algorithm [AL87]. In particular, it copes much better with curved objects. It produces results which are organized as three-dimensional chains of segments.

1 Introduction

We want to build a three dimensional description of a scene observed from several viewpoints. We need a description which is dense enough to allow an analysis of the scene and, in particular, makes it possible to identify some of the objects present.

We start with the results of a fast and accurate algorithm of trinocular stereovision [AL87]. We improve on this result by using figural continuity and local support. We also compute thredimensional results which are a good representation of the outline of the objects which are visible.

Our method has the following attractive features : flexibility (an automatic calibration technique is used to calculate precisely the intrinsic parameters and the position and the orientation of the cameras), density (in our experiments more than 50% of the image segments are matched), accuracy (less than 2% of the matches found are false), organized (the reconstructed 3D segments are structured as 3D-chains which represent a part of the outline of an object).

2 Edge Detection

The features of the image which are used by the stereovision algorithm are line segments. The basic idea of stereovision is to use some features which can be detected reliably in the image and match them between images taken from several viewpoints. Therefore we search, in the image, for some characteristic points : the edge points. They are usually defined as zero crossings of the laplacian [Gri84] or local maxima of gradient. Such points are generally the images of classes of significant physical attributes of the scene : surface discontinuities, shadows, markings on objects. These points can be computed efficiently and accurately and organized in edge chains [Der87],[Gir87]. An edge chain is a list of pixels corresponding to edge points. They do not have any triple point. From them, we build up a lists of line segments by using a polygonal approximation algorithm [Ber86]. These line segments are the features which are used by the stereo algorithms developed at INRIA [AL87].

Using line segments to represent image contours has the great advantage to reduce the amount of data to be processed and yields fast matching algorithms and very good results for scene containing mostly polyhedral objects. When curved objects are present, two problems occur which are caused by the unstability of the polygonal approximation.

Indeed, stereo algorithms can work only if the features used can be extracted from the images in a stable fashion : a real physical element, as a part of an object, must have similar attributes when viewpoint varies slightly. This is generally true for the edge points. Unfortunately the polygonal approximation algorithm is not always stable. This is especially true if the observed contour is curved. The matching of line segment is more difficult in this case.

When a match can be made, we want to reconstruct the 3D line segment which projects on the matched segments : we must select in each of them the parts which are common in terms of the epipolar strip. This part may be only a small component of the initial segment. In this case, we are unable to reconstruct a large part of the contour. As a further consequence the reconstructed 3D segments are not generally connected. This is an important problem, for later analysis of the 3D scene. To solve these two problems, we exploit the connectivity information which is present in the polygonal approximations of the edge chains. This is equivalent to the use of figural continuity.

3 Propagation using figural continuity

As a start, we use the matches found by an accurate and fast stereo algorithm [AL87]. This algorithm uses heavily the epipolar constraints and some compatibility constraints for the length and orientation of the matched segments. We propagate these matches by using figural continuity, i. e. the information provided by the edge chains.

This propagation can be easily achieved if we consider only segments in two images. We are not therefore obliged to restrict ourselves to the areas which are seen by the three cameras. However for reconstruction, we use once again the three images : since this decreases the uncertainty of the reconstructed segments.

To achieve this propagation, we apply rules and decide to match some segments or a part of a segment by scanning the chain to which their belongs.

Let us introduce some notations : $C^{i,a}$ is the i-th chain in the plane image of camera a. A chain is characterized by its number n of segments. We enumerate these segments from 1 to n. Segment p of chain $C^{i,a}$ is denoted $C_p^{i,a}$. A match, between two segments is noted $[C_p^{i,a}, C_q^{j,b}]$. For each chain, we say that the first segment is the head and the n-th segment is the tail.

As all epipolar lines intersect at the epipole, we can define a direction of propagation on the chain. For each segment on the chain, we call A its endpoint which is toward the head and B its endpoint which is toward the tail. The match $[C_p^{i,a}, C_q^{j,b}]$, defines an interval $[A_a, B_a]$ on segment $C_p^{i,a}$ and an interval $[A_b, B_b]$ on segment $C_q^{j,b}$. These intervals are computed using the epipolar constraint. Two solutions are possible A_a matches A_b or A_a matches B_b. In the first case, the directions of propagation for the match $[C_p^{i,a}, C_q^{j,b}]$ are the same for the chains $C^{i,a}$ and $C^{j,b}$. In the second case, the directions of propagation are inverted : if we walk on chain $C^{i,a}$ from the head to the tail, then we walk on chain $C^{j,b}$ from the tail to the head.

We can now define the following rules used in the propagation :

- Rule 1 (figure 1)

  Rule 1 deals with the case where we have two pairs of matched segments.

  If the following conditions are true. Segment $C_p^{i,a}$ matches segment $C_q^{j,b}$. Segment $C_{p'}^{i,a}$ matches segment $C_{q'}^{j,b}$. For each integer r between p and p', segment $C_r^{i,a}$ has not yet found any match

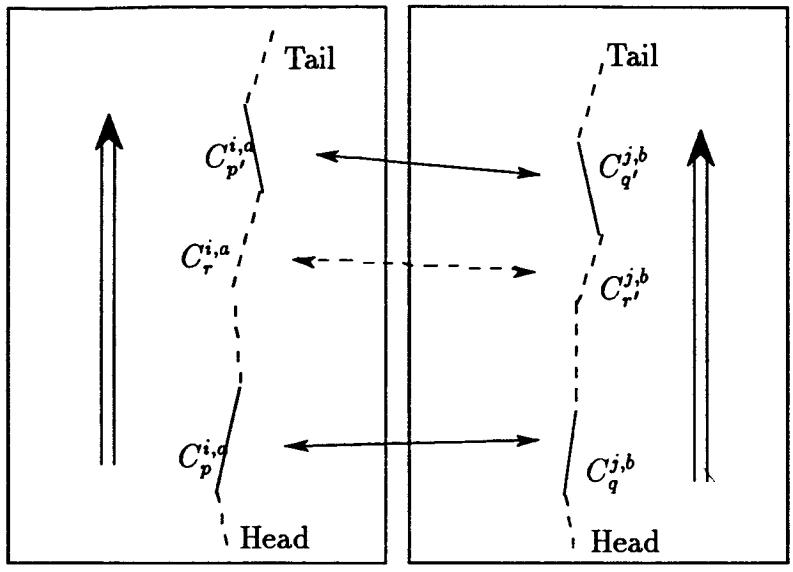


Figure 1: Propagation rule 1

in image b. For each integer s between q and q', segment $C_s^{j,b}$ has not yet found any match in image a.

Then, for each integer r between p and p', segment $C_r^{i,a}$ matches one of the segments $C_s^{j,b}$ of image b with s between q and q'.

- Rule 2 (figure 2)

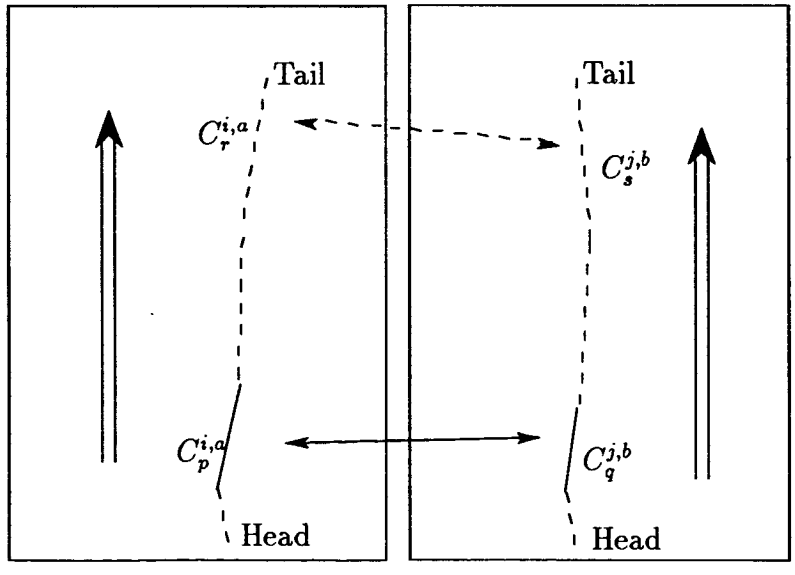


Figure 2: Propagation rule 2

This rules deals with the case where two segments match and all the segments from them to the end of the chain have not yet found any match.

If the following conditions are true. Segment $C_p^{i,a}$ matches segment $C_q^{j,b}$. For each integer r such that $r > p$, segment $C_r^{i,a}$ has not yet found any match in image b. The match $[C_p^{i,a}, C_q^{j,b}]$ defines a direction of propagation on the chain $C^{j,b}$. Two cases are possible but they are analogous. With respect to the match $[C_p^{i,a}, C_q^{j,b}]$, and the direction choose on the chain $C^{i,a}$, we consider that the segments following the segment $C_q^{j,b}$ are the segments $C_s^{j,b}$ with $s > q$ and the condition is : for each integer s, with $s > q$, the segment $C_s^{j,b}$ has not yet found any match in image a.

Then, for each integer r such that $r > p$, segment $C_r^{i,a}$ matches one of the segments $C_s^{j,b}$ with $s > q$.

This rule allows us to match the small segments, which sometimes are near the endpoints of chains.

- Rule 3 (figure 3)

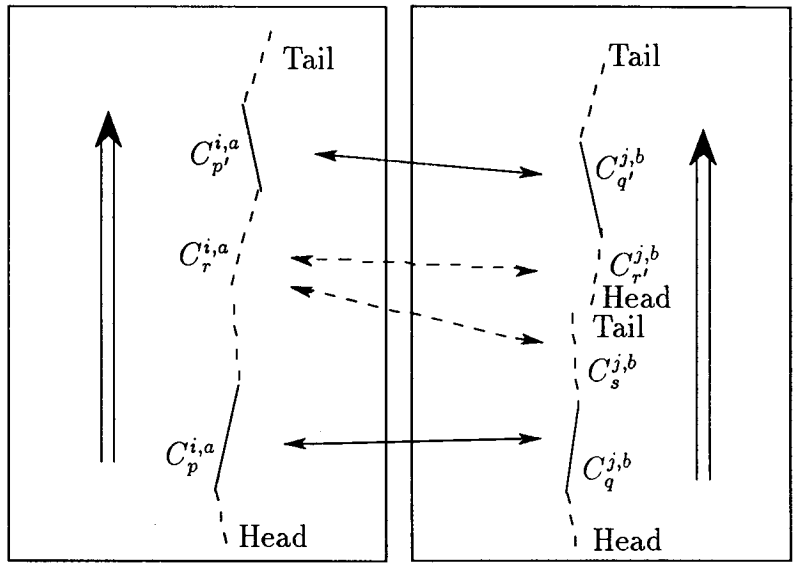


Figure 3: Propagation rule 3

This rule deals with the case where the matching chain of a chain has been cut in two chains in the opposite image by the segmentation process.

If the following conditions are true. Segment $C_p^{i,a}$ matches segment $C_q^{j,b}$. Segment $C_{p'}^{i,a}$ matches segment $C_{q'}^{j',b}$. To simplify, we suppose that $p < p'$ and that the epipolar constraint implies that the segment following the match $[C_p^{i,a}, C_q^{j,b}]$ are the segments $C_s^{j,b}$ with $s > q$, and the segment before the match $[C_{p'}^{i,a}, C_{q'}^{j',b}]$ are the segments $C_s^{j',b}$ with $s < q'$. All the possible cases are analogous. For each integer r between p and p', segment $C_r^{i,a}$ has not yet found any match in image b. For each integer s such that $s > q$, segment $C_s^{j,b}$ has not yet found any match in image a. For each integer s such that $s < q'$, segment $C_s^{j',b}$ has not yet found any match in image a.

Then for each integer r, between p and p', segment $C_r^{i,a}$ matches one of the segments $C_s^{j,b}$ with $s > q$ or one of the segments $C_s^{j',b}$ with $s < q'$.

With this rule, we have a way to correct some errors in the chaining process.

These rules define how line segments can be broken up into subsegments which are matched in the other image. We recursively compute them.

We begin with the match $[C_p^{i,a}, C_q^{j,b}]$ used to activate the rule. We compute the first interval which follows this match on the chain $C^{i,a}$. We calculate the epipolar strip of this part in the image b and take the intersection with the chain $C^{j,b}$. This part is not generally a complete segment. Therefore we must divide the segment in several parts which do not overlap. We implement the unicity constraint of the image of a 3D feature . We have thus created a new match which we use to continue this process.

This technique has the disadvantage of sometimes creating small partitionings on the segments. It therefore gives us some problems when we reconstruct the 3D segment.

These rules are applied to all pairs of cameras x, y. Some contradictory situations are possible with the third camera. This is uncommon because the epipolar constraint gives a very precise information. The contradictions are then solved during the reconstruction process.

A further unusual case is provided by the closed chains. We solve it by giving them an infinite length : the segment $(n+1)$ is the segment 1 if the chain has n segment. The rules, that we have defined, can then be applied without difficulty.

This propagation is computed very rapidly and the rules give very few false matches because edge chains are nearly always similar created in the different images and the matches used to begin the propagation are very accurate.

4 Matching using neighbourhood

Unfortunately, with the previous technique, we cannot define matches for the chains for which no match was initially found.

Therefore, we have developed a technique to describe the local environment of a segment. We first find the segments which are nearest to the segment $C_p^{i,a}$. We organize them into several classes. They are close to parallel to segment $C_p^{i,a}$. They form a T-shape with segment $C_p^{i,a}$. They have an endpoint near to one of $C_p^{i,a}$.

For a segment $C_p^{i,a}$, which has not yet found any match, we consider especially the segments which already match one segment in image b. Let us imagine that for example, segment $C_{p'}^{i',a}$ is a neighbour of segment $C_p^{i,a}$, and that segment $C_{p'}^{i',a}$ matches segment $C_{q'}^{j',b}$.

We define the set of segments of image b, which are candidates to be matched with the segment $C_p^{i,a}$, as all the segments $C_q^{j,b}$ which intersect the epipolar strip of the segment $C_p^{i,a}$ of image b. Among these, we look for all the segments $C_{q'}^{j',b}$ which are neighbours of segment $C_q^{j,b}$ and have the same position with respect to $C_q^{j,b}$ as segment $C_{p'}^{i',a}$ with respect to segment $C_p^{i,a}$.

When we explore all the segments which are neighbours of the segment $C_p^{i,a}$, we find, where possible, one or several segments, which are likely matches for segment $C_p^{i,a}$. We test if we have found enough neighours of segment $C_p^{i,a}$ and we test if the different candidates are compatible : the intersection of their epipolar strip with segment $C_p^{i,a}$ do not overlap. If these tests are false, we reject the match.

This matching technique requires a lot of computation to find the neighbours of a given segment. To decrease the search area, we organize the data. The image is divided in buckets and for each of

these buckets, we create the list of all the segments which intersect it. When we search for neighbours, we need only to explore the lists of the buckets near the segment currently being processed.

5 3D Reconstruction

3D reconstruction is one of the most important problems for stereovision algorithms. Our aim is to create spatial segment chains which are connected like the edge chains in the images.

Let us suppose that we match a set of m points M_i in m images. These points are the endpoints of part of the segments defined using the epipolar constraint. The coordinates of M_i in the image are : (u_i, v_i), for $i = 1, \ldots, m$. The coordinate $X = [x, y, z]^t$ of the spatial point M, whose images in the cameras are the points M_i are found by solving the equations :

$$\begin{cases} l_1^i X - u_i l_3^i X + l_{14}^i - u_i l_{34}^i &= 0 \\ l_2^i X - v_i l_3^i X + l_{24}^i - v_i l_{34}^i &= 0 \end{cases}$$

for $i = 1, \ldots, m$, ([FT86]). We have therefore a system of $2m$ linear equations in the three unknowns x, y, z which, in the exact case, has a unique solution. We can solve it in the real case by Kalman filtering which allows us to take into account both the noise on the images coordinates (u_i, v_i) and the result of calibration ([AF87]). We assume that the noise is gaussian. This noise appears with the digitization process, the edge detection and the polygonal approximation. The Kalman filtering yields a best estimation of X and its covariance matrix Λ.

Let us consider two spatial points M and P. We have computed them using Kalman filtering. We need to decide if they correspond to the same physical points. We call their covariance matrices Λ_M and Λ_P.

We assume that M and P are two independent gaussian points. Therefore the covariance matrix of the gaussian vector MP, is the sum $\Lambda_M + \Lambda_P$. If M and P are two instances of the same gaussian point MP, then the expected value of MP is 0 and the quantity

$$d^2(M, P) = MP^T(\Lambda_M + \Lambda_P)^{-1}MP$$

has a χ^2 distribution with 3 degrees of freedom (supposing that all points are gaussian). This quantity is the Mahalanobis distance. If $d^2(M, P)$ is less than some threshold s, the points M and P have a probability p (computed from χ^2 tables) of being two instances of the same physical point and we can fuse them. It is easy to build the fused point N by Kalman filtering. The uncertainty about the point N is less than that of M and P.

Notice that the Mahalanobis distance can be computed easily even though we must invert a 3×3 matrix. In fact the covariance matrix is symmetric and it is so possible to explicitly compute the expression of $d^2(M, P)$.

Thus we have a powerful tool for deciding to fuse the endpoints of a reconstructed segment. This allows us to build connected chains.

Indeed, let us consider a chain $C^{i,a}$ of image a. We reconstruct successively all the segments for this chain, if they are matched. If we find a match $[C_p^{i,a}, C_q^{j,b}]$, we also try to find the match with the third image c. It is the match $[C_p^{i,a}, C_r^{k,c}]$ and $[C_q^{j,b}, C_r^{k,c}]$. These last two matches do not always exist because they can be the image of a 3D segment which is not seen by the three cameras. We compute the endpoints of the 3D segment as described previously. For two segments $C_p^{i,a}$ and $C_{p+1}^{i,a}$, which follow each other on the chain $C^{i,a}$, we can decide using the Mahalanobis distance, to fuse the endpoints which are neighbours.

This allows us to build connected spatial segment chains which represent parts of an object. These chains can also be used for surface fitting algorithms or for object recognition and localization ([SS87]).

		doll	satellite
Segment Number	Cam 1	736	241
	Cam 2	836	217
	Cam 3	552	287
Segment Length	Cam 1	9600	6746
	Cam 2	9315	5278
	Cam 3	6822	7033
Initial Matches	Number	17.7%	27%
Initial Matches length	Cam 1	42.6%	31.1%
	Cam 2	41.1%	39.0%
	Cam 3	45.7%	29.2%
All Matches found	Number	63.3%	75%
All Matches found length	Cam 1	68.1%	48.4%
	Cam 2	66.3%	61.86%
	Cam 3	58.8%	45.72%

6 Experimental Results

Our algorithm is implemented in C.

We have tested it on a variety of images. Here we show the doll image (figure 5) and the satellite image (figure 7). For each scene, we show the polygonal approximations of the edge chains in the three images and the matches found, by projecting them back, on planes.

In the result of our algorithm, we notice a considerable improvement as for example on the head of the doll and the digits 5 in figure 5.

For each figure we give the number of segments for each image, the length, in pixels , of the segments in each image, the number of initial matches (it is a rate for the number of segments in the image of the first camera), the length of the parts of segments which are reconstructed in each image when using only the initial matches (it is a rate for the length, in pixels , of the segments in each image), the number of all matches found, (it is a rate for the number of segments in the image of the first camera), the length of the parts of segments which are reconstructed for each cameras when using all the matches. (it is a rate for the length, in pixels , of the segments in each image).

First we can note that the number of segments which are matched is considerably increased by our algorithm. In fact all the small segments are not matched by the algorithm used to find the first match because it is to difficult to find an accurate match for them among the three camera. With our algorithm it is now possible to take into account the local information and so we can match them using the information provided by the match of a longer segment of the same edge chain.

Since we subdivide the segments, the most characteristic information for the comparaison between the two algorithm is the length of the parts of segments which are reconstructed.

We also show the 3D reconstructions. We project them to a plane, which is not the same as those of the camera image plane.

For the propagation phase, the computation is approximately 30 seconds on a SUN-3 Workstation. The matching technique using an analysis of neigbourhood takes between 2 and 3 minutes. The spatial reconstruction of the segments also takes several minutes and depends on the number of matches found.

We can now present a very simple example of the use of the spatial chain for finding some structure in the 3D data. The key idea is to scan a spatial chain and to apply a least square on the segments.

For example we search a plane by applying least square algorithm for each endpoint of the segments of the chain. It calculates the best plane. After we look for the point which is the more longer from this plane and we can do a threeshold on this distance to decide that some chains are in a plane. For example, in the case of figure 5, all the points in the grid, which formed in the background of the scene, we determine this plane. All these points are less than 3 mm away from this plane.

7 Conclusion

We have described a stereovision algorithm which is a mixture of binocular and trinocular stereo. The main points of our approach are the following :

- It gives three dimensional maps which are much denser than in the trinocular stereo case by exploiting figural and neighbouring continuity.
- The maps are represented with connected segment chains.
- The uncertainty of the data has been reduced.

References

[AF87] N. Ayache and O.D. Faugeras. Building a Consistent 3d Representation of a Mobile Robot Environment by Combining Multiple Stereo Views. In *International Joint Conference on Artificial Intelligence*, August 1987.

[AL87] N. Ayache and F. Lustman. Fast and Reliable Passive Trinocular Stereovision. In *First International Conference on Computer Vision*, June 1987.

[Ber86] M. Berthod. Approximation polygonale de chaînes de contours. Programmes C, 1986. INRIA.

[Der87] R. Deriche. Using Canny's Criteria to Derive an Optimal Edge Detector Recursively Implemented. In *The International Journal of Computer Vision*, pages 15–20, April 1987.

[FT86] O.D. Faugeras and G. Toscani. The Calibration Problem for Stereo. In *Proceedings of CVPR'86,Miami Beach,Florida*, pages 15–20, 1986.

[Gir87] G. Giraudon. *Chaînage efficace de contour.* Rapport de Recherche 605, INRIA, Février 1987.

[Gri84] W. Eric L. Grimson. *Computational Experiments with a Feature Based Stereo Algorithms.* A.I. Memo 762, MIT, January 1984.

[II86] M. Ito and A. Ishii. Three-view stereo analysis. *IEEE Transactions on PAMI*, 8:524–531, july 1986.

[SS87] J.T. Schwartz and M. Sharir. *Identification of Partially Obscured Objects in Two and Three Dimensions by Matching of Noisy 'Characteristic Curves'.* Technical Report, Current Institute of Mathematical Sciences, New York University, 1987.

Figure 4: Scene with a doll : polygonal approximation

Figure 5: Reconstruction for the scene with a doll :Two projections planes are presented.

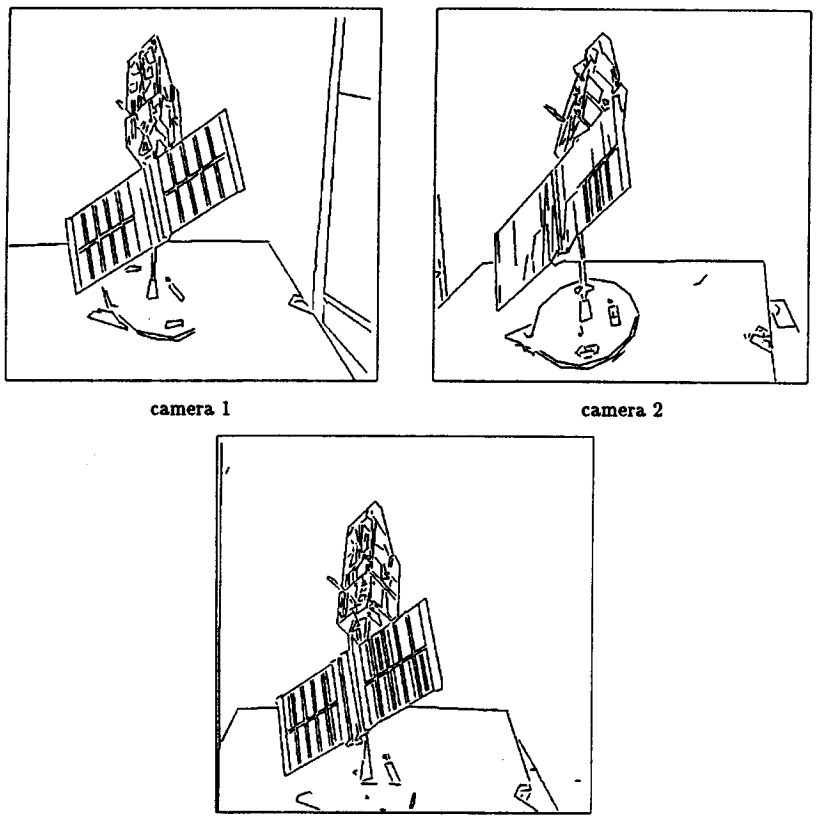


Figure 6: Satellite : Spot Polygonal Approximation

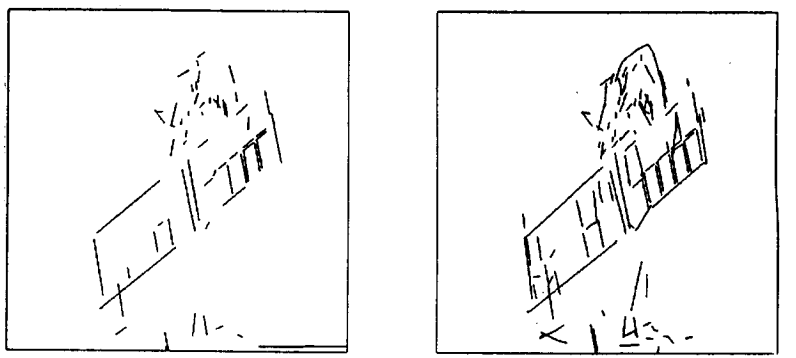


Figure 7: Reconstruction : satellite spot

A Fast 3-D Object Recognition Algorithm for the Vision System of a Special-Purpose Dexterous Manipulator

Stephen H.Y. Hung

National Research Council of Canada
Ottawa, Ontario, Canada K1A 0R6

Abstract

A fast 3-D object recognition algorithm that can be used as a quick-look subsystem to the vision system for the SPDM is described. Global features that can be easily computed from range data are used to characterize the images of a viewer-centered model of an object. This algorithm will speed up the processing by eliminating the low level processing whenever possible. It may identify the object, or reject a set of bad data in the early stage, or create a better environment for a more powerful algorithm to carry the work further.

1. Introduction

As Canada's contribution to the International Space Station, the Mobile Servicing System (MSS) is being developed to support a number of major functions on the Station, including Station assembly and maintenance, attached payload servicing, transportation, payload handling, and astronaut extra-vehicular activity. Many of the functions to be performed by the MSS require the execution of complex dexterous manipulation operations beyond the capability of the Space Station Remote Manipulator System (SSRMS). This functional capability is provided by the Special Purpose Dexterous Manipulator (SPDM). The SPDM consists of a base, body, and two manipulator arms. The manipulator arms are of the order of 1.5 to 2.0 metres long, have seven rotary joints each, and terminate with a tool changeout mechanism (Fig. 1). The SPDM is intended to be operated primarily from the end of the SSRMS. Control of the SPDM will be effected by the operator through one of the MSS work stations. Initially, the SPDM will operate in a teleoperated fashion with some autonomous capability. To reduce the crew time required for servicing and maintenance operations, increased degrees of autonomy will be incorporated in the SPDM operations as the system evolves. Detailed description of the SPDM can be found in Ref. 1.

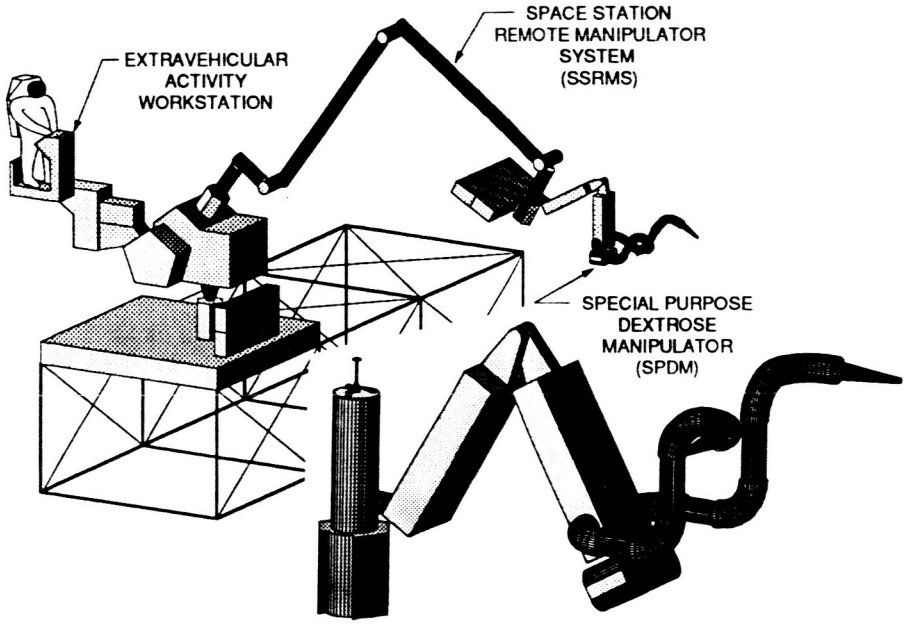


Figure 1.

The degrees of autonomy of the SPDM depend crucially on the vision capability. Vision sensors (video cameras, laser scanner) will be mounted near the manipulator arm ends. A vision system will also be mounted on the body to provide overall views of the work area of the manipulators. The vision systems will be able to identify objects (initially by reading labels), and to determine the position, the attitude, and rates of motion of a body relative to a known reference frame. Software and hardware provisions are being designed into the vision systems to enable enhancement of their capabilities to include more general pattern recognition and understanding of three-dimensional images. Three dimensional image data will be used in object identification and to update world models of objects for collision detection and avoidance. The data will be acquired with the stereo cameras or possibly a laser range finder. The algorithm presented in this paper is one of the possible methods designed for such applications.

2. Some Considerations

2.1. The space environment is characterized by very harsh lighting conditions, pronounced shadows, peculiar behaviour of light scattering, and the absorption bands in the near-infrared region of the earth background. All these are difficult for vision systems to handle. Thus, space vision systems should provide a mode independent of sunlight and be earth blind. Our experience suggests that the simplest approach would be to use laser range finders to collect range data for the space vision systems. Within the reach limit of the SPDM, laser range finders will work very well.

A laser range finder [2] has been under development in our laboratory for some years and has been used to collect a large quantity of range data [3]. It can capture a scene of 256 by 256 pixels within a second. Another compact wrist-mounted laser scanner [4] has also been successfully developed in our laboratory and has been mounted on the wrist of a PUMA robot arm to perform parts acquisition [5,6]. This compact scanner provides a single raster scan of range data. The robot arm must move in steps of small intervals to obtain the data for an area. Further development will allow range data over an area to be obtained by sweeping the wrist without moving the arm. Both of these systems are quite suitable for the vision system of SPDM.

The data obtained by both scanners have been used as the input data for extensive research work carried out in our laboratory on 3-D object recognition [7-13]. The algorithm proposed in this paper is based on the combination of some of these previous results.

2.2. In most space scenarios, the objects being viewed are unlabelled objects from a library of known and precisely defined objects. In our algorithm, we assume this to be the case and attempt to capitalize on it to reduce the computational effort of the space vision systems. Thus, we are adopting a model-based approach.

Model-based object recognition requires matching features measured in an observed image with models of objects. When 3-D objects may appear at any position and orientation, 3-D object models must be used. Usually, objects are modelled using either one object-centered representation or many viewer-centered descriptions (one for each viewpoint) [14]. In 3-D object-centered representations, objects are modelled by volumetric or surface models. Three-dimensional multiview representations model objects by a finite set of viewer-centered descriptions [15,16]. The major advantage of the multiview viewer-centered model over the object-centered representation is that the features extracted from images can be directly matched with the features associated with each viewpoint of the multiview model set.

However, the traditional approaches of either object-centered or viewer-centered models in object recognition have a common bottleneck. Both of them must extract local features such as edges, corners, surfaces, and the relations between them as basic data to describe an object or a class of objects in the reference data set. The major drawbacks of this kind of approach are the difficulty and large computational effort required in extracting the local features and their relations, as well as the complicated processing of the comparison. The combination of these drawbacks usually makes such methods slow and difficult to implement in realtime and, in some cases, they lead to a combinatorial explosion.

The algorithms based on the traditional approach may be powerful, but should not be used unnecessarily. There are many cases in which the use of these methods is obviously not desirable or at least should be postponed. The following are some of those cases:

A. If the observed object is obviously different from the object we are looking for, then the observed data should be rejected immediately without going through all the complicated low level processings.

B. The data are obtained from a poor viewing angle, and there is not enough information to identify the object. Such data should be discarded before the low level processing is involved and the system should be directed to look from another angle.

C. In many cases an object can be identified easily by looking at it from different viewpoints and relying only on some obvious global features without complicated low level processing.

This suggested that there should be a quick-look step before the low level processing to determine whether the full scale processing is actually needed. If the full processing is needed, then the quick-look step should be used to make sure that bad viewing angles have been avoided, and possible candidates have been narrowed down by eliminating those obviously unsuitable ones. This is true for all vision systems; it is especially true for the space vision systems. We should use a quick-look step to minimize effort and speed up the processing.

However, any such quick-look algorithm without low level processing to extract local features must rely only on the global features. The limitation of such approach is known. We must be able to separate the potential object either from the background or from the other objects.

2.3. The SPDM will be working autonomously in a space of about a cube of 2 metres in each dimension due to the reach limit of the arms. In such a space, the objects that SPDM will deal with can be roughly divided into three categories:

A. The object is isolated or can be isolated easily, such as a part or a tool flowing in the space.

B. The object is attached to a big object, and must be separated from the background, such as the ORU.

C. Two or more objects are stuck together, one may be over the other, and may or may not be separated physically.

The first two cases can be handled easily and should not be a problem. We must derive some kind of procedures to handle the case C. The most intuitive one is examination from a different angle, locating the top-most object, and slicing it off from the rest. In the space and the use of range data actually make such attempt easier.

In the following sections we will describe a quick-look algorithm and how to make it work. The main principle is that each step in the system is trying to create a better environment for the next step, until the object is identified.

3. Basic Assumptions and Brief Outline of our Approach

We assume that a laser range finder is mounted on the end of each arm and in the top of the lower body. The range finders can be programmed to scan a single profile or full resolution of 256 by 256 pixels. A reference plane can be set at any distance along the Z-axis. The readings of X, Y and Z are given in millimetres from the reference coordinate system at which the Z values of the reference plane are all zeros (Fig. 2).

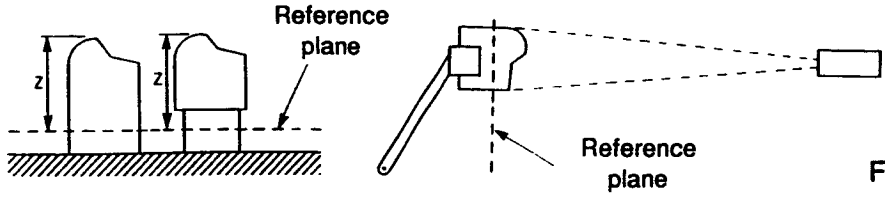


Figure 2.

The whole system can be divided into three subsystems:

3.1. The first stage is to locate the object and to determine the viewing angle and the reference plane. At this stage, we are using only the range finders in the single profile mode. They may need to scan several times to locate the object.

3.2. The second stage is a quick-look step. This step must be fast, not involving any low level processing, and able to be repeated easily without much delay to the whole processing. This subsystem should be able to eliminate the most obvious unsuitable candidates and determine if the input data can be identified, be rejected, or if this quick-look processing should be repeated by looking from another angle, or if the secondary subsystem should be invoked.

3.3. The secondary subsystem. When the quick-look subsystem fails to give a conclusive answer, a more powerful subsystem must be called in to carry the processing further. In this stage, low level processing will be involved, and features will be extract. Any traditional or otherwise method can be adopted as one see fit. The processing may be more complicated, but will only be invoked when necessary, and under the more favourable conditions prepared by the quick-look subsystem. This stage is beyond the scope of this paper and will not be discussed further.

4. The Algorithm for the Quick-look Subsystem

The algorithm for the quick-look step in this system is described in [12]. It is based on multiview viewer-centered representation. Each image of an object from a possible viewing angle is characterized by four global features which can be quantified and can be readily calculated from the raw data of the image. The features of images from all possible viewpoints of an object are stored as the model of the object. The collection of all models makes up the library of models for the quick-look subsystem.

The four features chosen are the approximate size and shape of the silhouette, the histogram of Z-values and the distribution of Z in the (x,y) plane.

4.1. The size of the silhouette is considered as the most obvious feature to distinguish an image of a particular viewing position of an object. Since the scanner is making a reading of Z at a constant interval along both the X and Y axes, the size of a silhouette is simply the number of points in an image. It is invariant under in-plane rotation and translation.

4.2. The shape of the silhouette is the next obvious feature to be considered. A moment invariant is chosen to represent the shape of the silhouette. It is the IV2D below:

$$\phi_{pq} = \sum_x \sum_y (x-\bar{x})^p \cdot (y-\bar{y})^q \quad \text{(central moment)}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of x and y, respectively

$$\psi_{pq} = \phi_{pq}/\phi_{00}^{\gamma}, \text{ where } r = \frac{1}{2}(p+q)+1 \qquad \text{(normalized central moment)}$$

$$\mathbf{IV2D} = (\psi_{20} + \psi_{02}), \quad \mathbf{P_2 = IV2D \times 1000.0}$$

Since the value of IV2D generally is too small, we multiply it by 1000.0 and then use it as a parameter P_2. Parameters P_2, P_3 and P_4 are independent of size, in-plane rotation and translation. P_2 is solely determined by the shape.

4.3. The histogram of Z-values is a good indicator of the volatility of the visible surfaces. The best parameter to describe the histogram is its standard deviation, i.e.,

$$\sigma = \sqrt{\frac{\sum\sum_{xy} (z - \bar{z})^2}{N}}, \quad z = f(x,y) \qquad \text{where: } \bar{z} \text{ is the mean value of } z = f(x,y),$$
$$N \text{ is the number of points.}$$

The standard deviation is also independent of the height of the object. The standard deviations of two different objects can be compared without knowing the heights of the two objects.

4.4. The distribution of Z-values in the (x,y) plane is a good descriptor of the visible portion of a 3-D object at a certain viewing angle, i.e., a 3-D image. The quantity used to represent the distribution of Z is an invariant of moments indicated as IV3D:

$$\mu_{pq} = \sum\sum_{xy} (x - \bar{x})^p \cdot (y - \bar{y})^q \cdot f(x,y), \quad z = f(x,y) \qquad \text{(central moment)}$$

$\bar{x}$ and $\bar{y}$ are the mean values of x and y, respectively

$$\eta_{pq} = \mu_{pq}/\mu_{00}^{\gamma}, \text{ where } \gamma = \frac{1}{2}(p+q)+1 \qquad \text{(normalized central moment)}$$

$$\mathbf{IV3D} = (\eta_{20} + \eta_{02}), \quad \mathbf{P_4 = IV3D \times 1000.0}$$

P_4 is defined as IV3D multiplied by 1000.0. P_4 is not independent of the Z-values. The comparison of P4 of two images will be meaningful only if the two images are the same distance from the reference plane. The P_4 at a certain height can always be computed from the P_4 at a different height, provided that the average value of Z (AVZ) is also available. Thus, the P_4 can always be compared at the same height.

The four parameters are:

P_1 = Size of the Silhouette (Number of Points)
P_2 = Shape of the Silhouette (IV2D × 1000.0)
P_3 = Volatility of the Visible Surface (σ, Standard Deviation of the Z-values)
P_4 = Distribution of Z-values in (x,y) plane (IV3D × 1000.0)

For example, the four parameters of the grapple feature when scanned directly from the top centre are shown in Fig. 3. Because this set of data includes the round platform, this set of parameters does not really show the special character of this feature. Since the highest point in the data or the distance to the platform is known, the platform can be sliced off and a set of data obtained as in Fig. 4. The parameters computed from this set of data give a better representation of the grapple from this particular viewpoint.

One of the important reasons for choosing these four parameters is that all four parameters and the average value of Z (AVZ) can be readily computed by going through the data only once. Thus, low level processing is not needed. The first three parameters are used to classify the images of objects in a three-dimensional feature space. Once the four parameters of the input data are obtained, an allowable error limit is set for each of the first three parameters. The input data are characterized by a small cube in the feature space in such a way that the centre of the cube is determined by the first three parameters, and each dimension is the allowable error. The search is simply to determine if this cubic actually intersects any clusters (characterized images). If there is no intersection, then the input data are rejected. Otherwise, the possible candidates and the input data will be adjusted to have the same AVZ; then the P_4 will be compared.

If the result is inconclusive, then examination from another viewing angle will proceed. Usually the second or the third try will give a definite answer (identified, rejected, or secondary subsystem should be invoked). If after several (five or six) tries there is still no conclusive answer, then the secondary subsystem should be invoked.

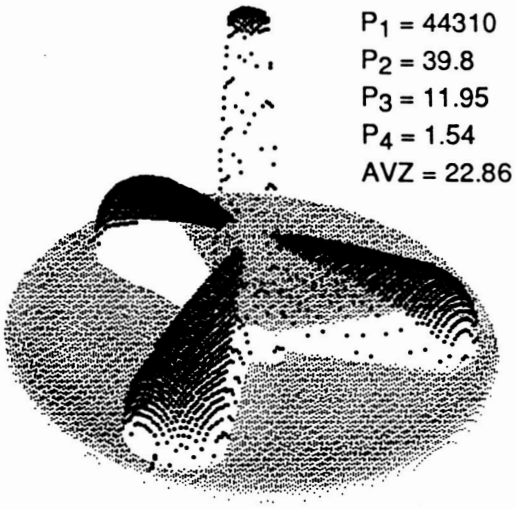


Figure 3.

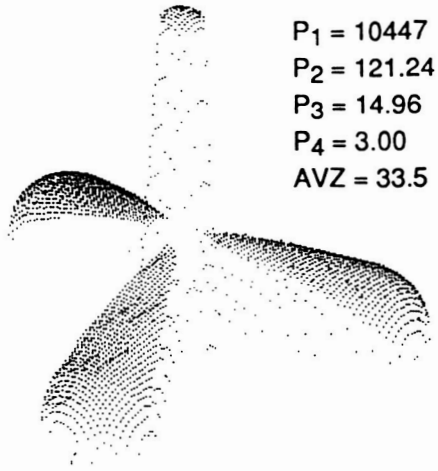


Figure 4.

5. The Searching Subsystem and the Secondary Subsystem

The major mandate of the searching subsystem is to locate the object. If there is something to look at, then the subsystem must determine the viewing angle, how to set the reference plane, and the approximate area to be scanned to cover the possible object.

The analysis of the single profile of the laser scanner can be used to determine if there is an object. If there is a possible object, then a few cross or parallel scans will be enough to decide the approximate area to cover the whole object. The viewing angle and the reference plane should then be set to separate the object from the background or other objects.

The rule of thumb is that whenever possible we should scan from an angle that is perpendicular to the major surface or the background, and the reference plane should be set to cross the jump edges in a single profile of range data, as in Fig. 5.

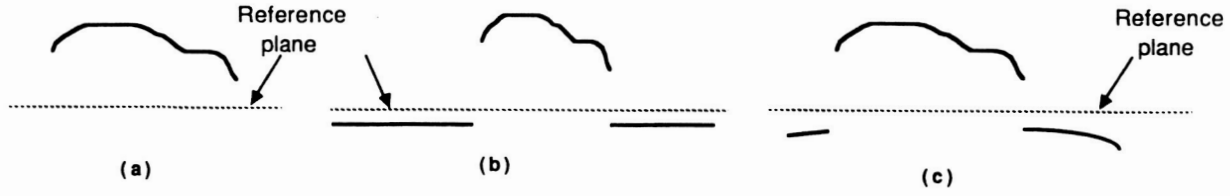


Figure 5.

When two or more objects stick together, the best we can do is to slice off the top-most object from the rest. In range data, the top-most object is usually separated with the rest of the scene by the jump edges. The object at the bottom will become the top-most one when looked at from a different angle, as in Fig. 6. If the two pieces are not stuck together physically, the bottom one may be identified by moving the top pieces away first. This stage should provide the quick-look subsystem with a set of workable data.

A separated portion in the model of each object is specially set aside for the searching subsystem. In this portion, major dimensions are listed and can be directly referred to without being extracted from the 3-D model.

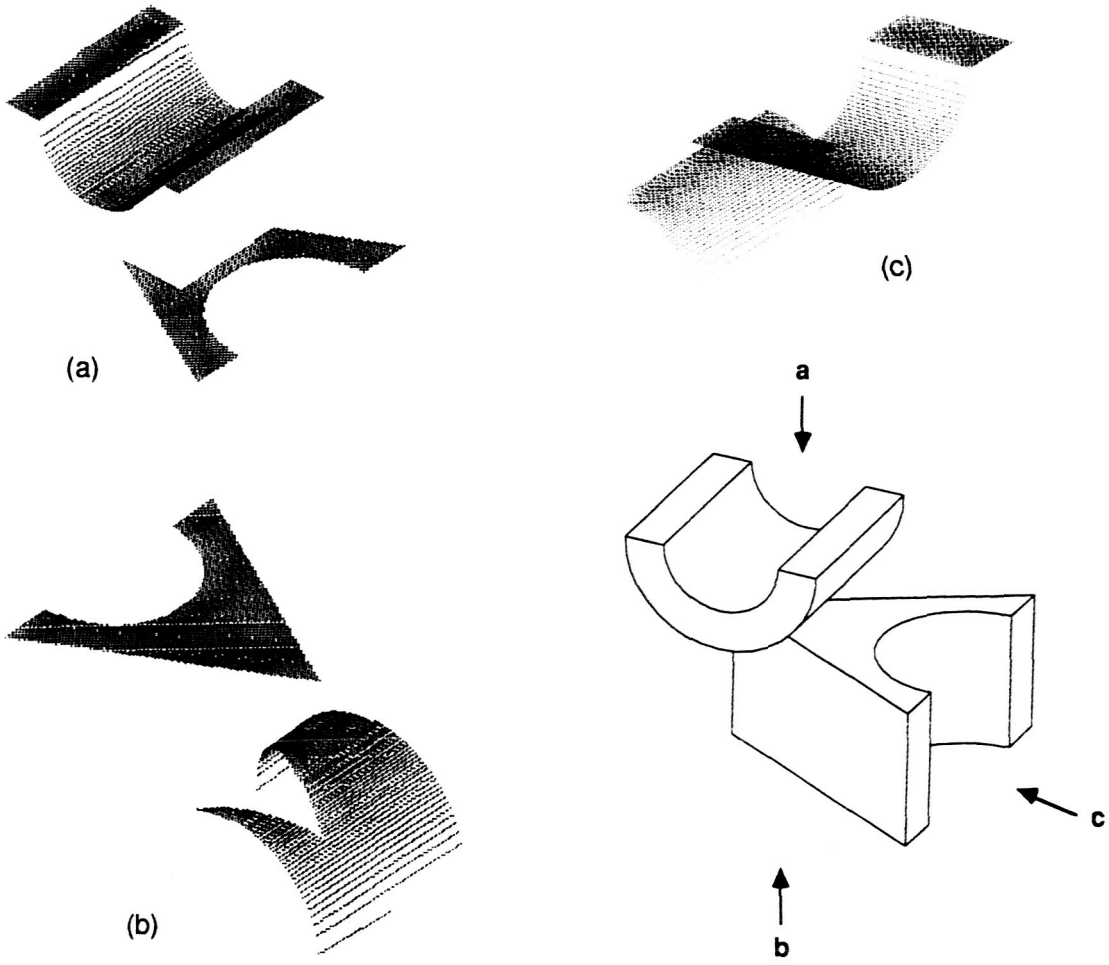


Figure 6.

A few points can be made about the secondary subsystem:

A. In most cases, the secondary subsystem is called to narrow down several possible candidates to a single one. Since the observed object is known by now to possibly be identified with one object in a group of similar objects, a tailored algorithm may be enough to handle the situation. Extracting of some special features from the 3-D model will provide everything needed to do the job. Thus, a CAD-based representation may be preferred to a volumetric representation, although the latter may make it easier to compare two complete objects.

B. A multiview representation in this stage may be neither necessary nor desirable. The model for the quick-look subsystem is a multiview representation in a simpler form. The possible advantage of the multiview model is already taken. The ease of extraction of a special feature from a 3-D model is the most important concern at this stage, the object-centered model seems better in this respect.

C. The most frequent reason for the failure of the quick-look subsystem is that the object has not been properly separated from the scene. Before the full scale of low level processing is involved, a limited low level processing which extracts only the jump edges can be applied. The result may help separate the top-most object from the rest of the scene, and conclusive result may be obtained by applying the quick-look steps again.

The model of an object for the system is shown in Fig. 7. Besides the information described previously, there is a place for some special instructions in each stage. At the beginning, the special instructions are put in by the system designer. In the future, the system will have a self-improving ability to update those instructions as the system learns from its experience. This may be in a distance future, but it is a must for a vision system to work properly.

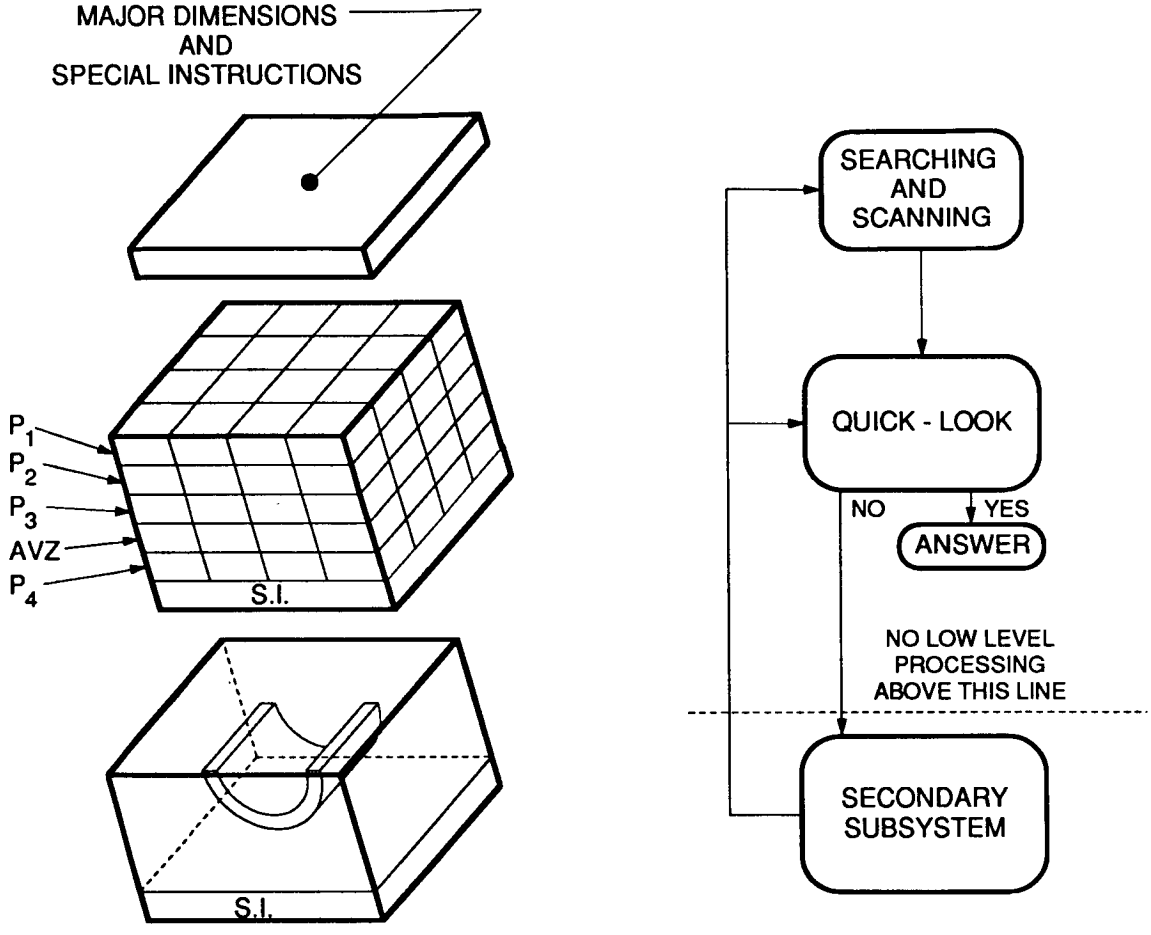


Figure 7.

6. Example

The procedure for locating and confirming the grapple feature is described as follows:

1. Assuming that the grapple feature is located in the end of a large cylindrical body. The SPDM is guided by the SSRMS to the vicinity of the large body. The searching subsystem must determine the orientation of the body and lead the SPDM to one of its end by analyzing some single profile data (Fig. 8).

2. When the SPDM is at one of the ends of the body, at least two more single profile data must be obtained to determine whether the SPDM is at the top centre and perpendicular to the platform of the end of the body. The reference plane is set to just beyond the platform.

3. After scanning is completed, the set of range data is read once and the four parameters are computed. If the SPDM is at the right end, then the parameters should be closed to the set in Fig. 3, provided the viewing angle is not far off the top centre (within few degrees).

4. This situation should be further confirmed by slicing off the platform and computing the parameters for the remaining data. The parameters should be similar to the set in Fig. 4. This will also confirm that the SPDM is viewing from the top centre and perpendicular to the platform. Otherwise, the parameters computed from the remaining data (Fig. 9) after the slicing off action cannot match the set in Fig. 4.

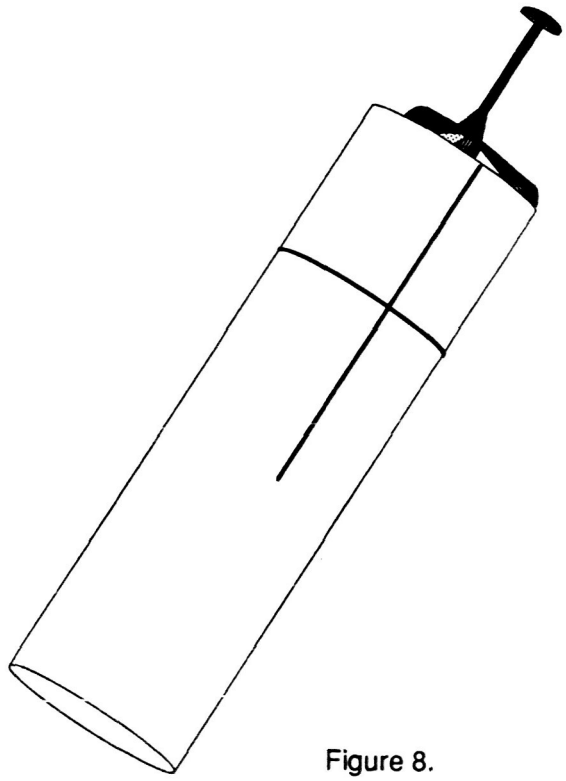


Figure 8.

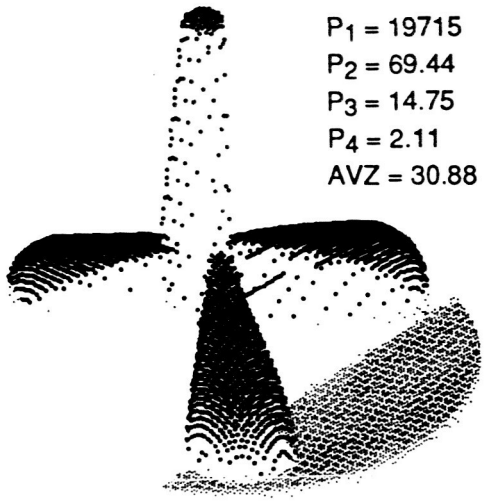


Figure 9.

7. Conclusion

A vision system has been proposed for the SPDM. Range data are chosen as the input data for this system. The emphasis is on a quick-look step that should be implemented before the low level processing becomes involved. By considering four aspects of the images of an objects in range data, the quick-look algorithm can speed up the processing by identifying the object, rejecting the data in the early stage, or reducing the number of possible candidates remaining in the field. A search step is applied to pave the way for this quick-look algorithm.

Major effort for implementing this quick-look algorithm is now being placed on the building of the library of models. This involves scanning the objects from many different angles to obtain the range data, computing the parameters, and then discarding the data and storing only the parameters. However, all these are done in off-line operation, little on-line computation will be required to obtain a conclusion when the input data are given.

This is only a preliminary study on the feasibility of a possible space vision system for the SPDM. Much must be done to make it practically worthwhile.

8. References

1. Hunter,D.G., "An Overview of the Space Station Special Purpose Dexterous Manipulator (SPDM)". National Research Council of Canada, NRCC No. 28817, April, 1988.

2. Rioux, M., "Laser Range Finder Based on Synchronized Scanners". Appl. Opt. 23(21), pp. 3837-3844; 1984. NRC 24529.

3. Rioux, M., and Cournoyer, L., "The NRCC Three-dimensional Image Data Files". CNRC 29077.

4. Rioux, M., Bechthold, G., Taylor, D. and Duggan, M., "Design of a Large Depth of View Three-dimensional Camera for Robot Vision". Optical Engineering, 26(12), pp. 1245-1250; 1987. NRC 29130.

5. Roth, G., and O'Hara, D.H., "A Holdsite Method for Parts Acquisition Using a Laser Rangefinder Mounted on a Robot Wrist". Proc. IEEE Int'l. Conf. on Robotics and Automation, Raleigh, North Carolina, March, 1987, pp. 1517-1523.

6. Archibald, C., "Processing Single Range Profiles from a Wrist Mounted Laser Range Finder". Proc. Vision Interface'88, Edmonton, Alberta, June, 1988, pp. 13-18.

7. Archibald, C., and Rioux, M., "Witness: A system for Object Recognition Using Range Images". NRC ERB-986, Jan. 1986.

8. Oka, R., Kasvand, T. and Rioux, M., "Recognition of Three-dimensional Objects Based on the Cross-angle Transform". NRC ERB-980, April 1986.

9. Dhome, M., and Kasvand, T., "Polyhedra Recognition by Hypothesis Accumulation". IEEE Trans. PAMI-9(3), pp. 429-438, 1987. NRC 29137.

10. Hospital, M., Yamada, H., Kasvand, T., and Umeyama, S., "3D Curve Based Matching Method Using Dynamic Programming". Proc. 1987 IEEE First International Conf. on Computer Vision, London, England. June 8-11, 1987. pp. 728-732.

11. Abdelmalek, N.N., "A Global System for matching 3-D Range Data Objects". Proc. Vision Interface'88, Edmonton, Alberta, Canada, June 6-10, 1988. pp. 129-134.

12. Hung, S.H.Y., "A Practical Approach of 3-D Object Recognition in Range Data". Proc. First Int'l. Conf. Industrial & Eng. Appl. of Artificial Intil. & Expert Systems. UTSI, Tulahoma, Tennessee, June 1-3, 1988. pp. 1131-1138.

13. Merritt, C., Archibald, C., and Ng, T., "Pose Determination of a Satellite Grapple Fixture Using a Wrist-mounted Laser Range Finder". Proc. SPIE 1002, Intelligent Robots and Computer Vision, Cambridge, MA. Nov. 1988.

14. Chin, R.T., and Dyer, C.R., "Model-Based Recognition in Robot Vision". Computing Surveys, Vol. 18, No. 1, pp 67-108. March 1986.

15. Ikeuchi, K., "Generating an Interpretation Tree from a CAD Model for 3D-Object Recognition in Bin-Picking Tasks". Int'l. J. Computer Vision, pp 145-165, 1987.

16. Kron, M.R., and Dyer, C.R., "3-D Multiview Object Representations for Model-based Object Recognition". J. Pattern Recognition, Vol. 20, No. 1, pp 91-103, 1987.

USE OF 3D VISION FOR FINE ROBOT MOTION

Anatole Lokshin and Todd Litwin

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove, Pasadena, Ca. 91109

I. Introduction

A wide use of robotic manipulators is handicapped by their inability to operate in an unstructured environment. The existing industrial systems operate by moving along pretaught trajectories, or in the best case rely on *a priori* CAD world models. Since these motions are "blind" any distortion in robot trajectory due to calibration errors can be catastrophic. Besides the obvious inflexibility of such approach, significant problems of robot calibration have to be solved in order to exercise these motions [Stone].

An integration of 3D vision systems with robot manipulators will allow robots to operate in a poorly structured environment by visually locating targets and obstacles. However, by using computer vision for objects acquisition makes the problem of overall system calibration even more difficult. Indeed, in a CAD based manipulation a control architecture has to find an accurate mapping between the 3D Euclidean work space and a robot configuration space (joint angles). If a stereo vision is involved, then one needs to map a pair of 2D video images directly into the robot configuration space. Neural Network approach [Kup88] aside, a common solution to this problem is to calibrate Vision and Manipulator independently, and then tie them via common mapping into the task space. In other words, both Vision and Robot refer to some common *Absolute* Euclidean Coordinate Frame via their individual mappings.

This approach has two major difficulties. First a Vision System has to be calibrated over the total work space. And second, the Absolute Frame, which is usually quite arbitrary, has to be the same with a high degree of precision for both Robot and Vision subsystem calibrations.

This paper describes a work aimed on the use of computer vision to allow robust fine motion manipulation in a poorly structured world which is currently in progress at the JPL along with the preliminary results, encountered problems, and directions for the future work.

II. System Setup Description

The JPL Telerobot Testbed, where the described work has been done, includes several subsystems working cooperatively. There are low level mechanization subsystem - Manipulation and Control Mechanization (MCM), Run Time Control (RTC) subsystem, Operator Control Station (OCS), AI Task Planner, and Sensing and Perception (S&P) subsystems. More detailed description of the Telerobot Testbed and individual subsystems can be found elsewhere [Matij],[Kan 1-2], [Stone-2]. Robot Vision is provided by the S&P subsystem which determines object 3–D positions and orientations by matching a pair of stereo images with prestored polyhedral models. A detailed description of the algorithms and special purpose hardware used by the S&P is given in [Gennery].

The Telerobot Testbed has five video cameras which are available for the Sensing and Perception subsystem (S&P). Three cameras are stationary and provide wide angle - F = 12.5 mm - view of the total workspace. Two cameras are mounted on a wrist of a Puma-560 manipulator, called Camera Arm, and can be used to provide a close view stereo (F = 25.0 mm). The stationary cameras are used mainly to acquire and track large objects such as satellites, and do not provide enough resolution for fine motion manipulation. Only the movable close view cameras can be used to provide positional information for the objects in a task space. Here we will be concerned only with the Puma arm mounted stereo cameras. Therefore the terms "cameras" or "vision" will assume only a pair of potentially movable cameras and not the three stationary wing cameras.

The movable cameras can be brought close enough to the Task Board to provide a good view of a particular object of interest, however the total work volume where an object can be focused in both cameras is only about 1'x1'x1'. As a result, any realistic Payload Servicing Task requires an ability to move the cameras and to perform machine-vision operations from arbitrary locations.

The Sensing and Perception (S&P) subsystem was designed around stationary cameras which have to be calibrated only once. Due to a very time-consuming nature of a camera calibration process the same approach cannot be used for the movable cameras. It is absolutely required, that a 2D $\rightarrow$ 3D mapping of the cameras be found only once at a particular configuration of the Camera Arm, and then could be recomputed for any arbitrary Camera Arm configuration. The following method allows such an alteration.

III. Extension of the Vision Work Space

The camera model used by the Sensing and Perception Subsystem is composed of four 3-vectors: c, a, h, and v, which are known as the center, axis, horizontal, and vertical vectors, respectively (Fig. 1). These vectors describe the relationship between the 3D coordinates of the Telerobot Reference Frame (used as Absolute Frame) and the 2D image coordinates of a stationary camera. They are derived from a rather extensive and time-consuming calibration process. As long as the camera is not moved from its calibrated position then these vectors are sufficient as they are. If, however, the camera needs to be moved, then the vectors of the camera model must be altered to reflect the movement.

For each camera, there is defined an arbitrarily placed camera reference point which is rigidly attached to the camera. While this additional coordinate frame can be set anywhere, to simplify computations this frame was coincided with the wrist origin of the Camera Arm. If this point is located in the Telerobot Reference Frame at position p (a 3-vector) and orientation q (a quaternion) before the camera is moved, then the camera model (c, a, h, v) can be transformed to a new camera model (c', a', h', v') at location (p', q') after the move by the following:

$$\begin{aligned} c' &= p' + R(c - p) \\ a' &= Ra \\ h' &= Rh \\ v' &= Rv \end{aligned} \qquad (*)$$

where

$$R = r(q')r^T(q)$$

and $r(q)$ is a function which transforms a quaternion q into a rotation matrix (Fig. 1).

However, a practical application of this algorithm leads to a significant mismatch between the real and S&P perceived absolute object positions.. The errors are about 2 - 4 cm and result from the combination of errors in the Puma Arm nominal kinematics, Vision calibration based on a fictitious absolute frame, and measuring errors in the abs positions of the task objects. These errors could be significantly reduced by using more detailed model for arm kinematics [Hayati] and by much more elaborate measurements for the Task Data Base. However, it is our opinion, that an application of the computer vision in robotics should lead to the inherently robust algorithms which provide an ability to work with nominally known systems

operated in an unstructured world. To achieve these results, we substituted the whole concept of Vision Based Absolute Motion, by a Vision Based Relative Motion.

The next section describes an approach which allows employment of moving cameras by using independent calibration of Vision and Manipulation Systems and without any significant improvements in the determination of p and p' absolute coordinates.

III. Absolute vs. Relative Robot Motions

Let us compare the accuracy in the positioning of a Robot End Effector (EE) in two cases. In one case the Robot is asked to reach a position P given in some Absolute Frame, say Telerobot Reference Frame (Fig.2). Alternatively, for a second case we will command the Robot to move its EE to a position P_1 which is known relative to the EE start location P_0 (Fig. 3).

We will use standard matrix notations to describe object positions and orientations in the standard $R^3 \times SO^3$ space [Paul]. Let Z be a transformation from the origin of the absolute frame to the shoulder of Puma manipulator, and T^{0P} gives transformation from the shoulder to some point P, say EE or camera reference point. For any true value Y we will denote its computed or obtained estimate as $\hat{Y}$.

A. Absolute Motion

Let us put the EE at absolute location P which is described in the abs frame of reference by a transformation $T^{EE}_{des}(P)$. The desired joint angles can be found (and are actually computed in the existing Telerobot Testbed) via inverse kinematics of $\hat{T}^{0E}(P)$, where

$$T^{EE}_{des}(P) = \hat{Z}\hat{T}^{0E}(P) \tag{1}$$

therefore

$$\hat{T}^{0E}(P) = \hat{Z}^{-1}T^{EE}_{des}(P) \tag{2}$$

A Nominal Inverse Kinematics (NIK) algorithm transfers $\hat{T}^{0E}(P)$ into a set of desired joint angles J_{des}, which are sent to the joint controllers. As a result, the arm moves to some set of joint angles J which is slightly different from J_{des}. Due to the imperfection of the NIK model and joint controllers the achieved location of the EE P_1 is different than the desired P. The resulting shoulder-EE transform $T^{0E}(P_1)$ is different from the commanded $\hat{T}^{0E}(P)$ by some error $D_{kin}(P)$ (if P and P_1 are close then $D_{kin}(P) \sim D_{kin}(P_1)$)

$$T^{0E}(P) = \hat{T}^{0E}(P)D_{kin}(P). \tag{3}$$

The final position of the EE becomes

$$T^{EE}_{abs}(P_1) = ZT^{0E}(P_1) \tag{4}$$

and by substituting (3) into (4) we have

$$T^{EE}_{abs}(P_1) = Z\hat{T}^{0E}(P)D_{kin}(P) \tag{5}$$

By substituting (2) and (1) in (5) we can find an absolute error in the EE positioning

$$T_{abs}^{EE}(P_1) = Z\hat{Z}^{-1}T_{des}^{EE}(P)D_{kin}(P) \tag{6}$$

If an estimate of the shoulder transform differs from its true value as

$$Z\hat{Z}^{-1} = D_Z \tag{7}$$

then the final position of the EE becomes

$$T_{abs}^{EE}(P_1) = D_Z T_{des}^{EE}(P)D_{kin}(P) \tag{8}$$

One can see from (8) that if $T_{des}^{EE}(P)$ involves large translational motions, which is usually the case, even a small rotational error in Z may lead to a significant absolute difference between the desired and actual position of the End Effector.

If T_{des}^{EE} is computed from vision or CAD data, it has its own error

$$T_{des}^{EE}(P) = \hat{T}_{abs}^{EE}(P) = T_{abs}^{EE}(P)D_{vis}(P) \tag{9}$$

and the final difference between the desired position P and achieved P_1 becomes

$$T_{abs}^{EE}(P_1) = D_Z T_{abs}^{EE}(P)D_{vis}(P)D_{kin}(P) \tag{10}$$

The experimental results with the PUMA 560 setup at the Telerobot Testbed showed, that a Puma arm with a carefully calibrated NIK can achieve about 5 mm accuracy. After the CAD Data Base was updated by touching a number of points on the Task Board with a special tool, an accuracy was improved to about 2 mm.

The use of the Vision System in a single, carefully calibrated configuration leads to about the same result, however, if a "moving camera model" described in sec. II is used, then the vision perceived object positions are about 20 – 40 mm off. Fig. 4 shows a Vision Overlay obtained with the stationary calibrated cameras. Fig. 5 shows performance degradation when the described above "Moving Cameras" method (eq. (*)) had been used.

B. Relative Motion

In the previous paragraph we have shown that the major potential source of the robot positioning errors is an uncertainty in the robot location relative to the "Absolute" coordinate frame.

Suppose now that we want to make a relative motion of EE from its current position P_0 to P_1 on some well known transformation A. Then the desired end position is

$$T_{des}^{EE}(P_1) = T_{abs}^{EE}(P_1) = T_{abs}^{EE}(P_0)A$$

An estimate of the desired position has to be based on the perceived current location of the EE

$$\hat{T}^{EE}_{des}(P_1) = \hat{T}^{EE}_{abs}(P_0)A \tag{11}$$

or from (1), (2), (3) and (11)

$$\hat{T}^{EE}_{des}(P_1) = \hat{Z}D^{-1}_{kin}(P_0)T^{0E}(P_0)A = \hat{Z}D^{-1}_{kin}(P_0)T^{0E}(P_1) \tag{12}$$

By substituting (12) into (6) we have

$$\hat{T}^{EE}_{abs}(P_1) = Z\hat{Z}^{-1}\hat{Z}D^{-1}_{kin}(P_0)T^{0E}(P_1)D_{kin}(P_1) = ZD^{-1}_{kin}(P_0)Z^{-1}T^{EE}_{abs}(P_1)D_{kin}(P_1) \tag{13}$$

We see from the (13), that if an arm has a good kinematics model then its relative motion can be done very accurately. But the most important feature of the derived relation (13) from the Vision – Motion coordination point of view is its independence on $\hat{Z}$. It shows that Cameras and Manipulator can be calibrated independently relative to their local frames. Namely, one needs to know only the manipulator's inverse kinematics and to be able to determine a true 3D relative position between two objects.

This property of (13) may be used for fine motion of Robot EE. Say a particular object has to be grasped by the EE. Suppose also, that the grasping should start from an approach position which is located A relative to the object. In other words the desired absolute position of the EE is

$$T^{EE}_{abs}(P_{apr}) = T^{obj}_{abs}A \tag{14}$$

If the current position of the EE is P then a relative motion B can move it to the P_{apr}, where

$$T^{EE}_{abs}(P)B = T^{obj}_{abs}A \tag{15}$$

Suppose now, that instead of the true *abs* positions in (15) one is using the vision derived values. Then the desired adjustment B is

$$\hat{B} = [T^{EE}_{cam}(P)]^{-1}T^{ob}_{cam}(P_{obj})A \tag{16}$$

Let $T_{abs}(P) = T_{cam}(P)D_{vis}(P)$ then from (16)

$$\hat{B} = [D^{-1}_{vis}(P)T^{EE}_{abs}(P)]^{-1}D^{-1}_{vis}(P_{obj})T_{abs}(P_{obj})A \tag{17}$$

If P is close to P_{obj}, which is usually the case, then $D_{vis}(P) \sim D_{vis}(P_{obj})$ (since obviously $D_{vis}(X) = D_{vis}(Y)$ when $X = Y$ and we expect the error to be a smooth function of $X - Y$) and A is small, eq. (17) gives a very good approximation to the relative distance B (vision error is uniformed across the field of view) then

$$\hat{B} = B \tag{18}$$

Eq. (18) together with (12) shows, that using Vision Based Relative Motion vision and manipulator calibration can be separated and performed in the relative sense only. Namely, instead of accuracy of absolute

motions only manipulators Inverse and Forward Kinematics – motions in the shoulder frame should be calibrated. Similarly, only relative camera calibration (relative to their field of view) is important. Such calibration can be done only once with a high precision calibration fixture and then eq. (*) can be used.

IV. Discussion

The Vision Based Relative Motion calls for the following possible scenario. Operator moves Cameras to position an object of interest in Cameras field of view. Then the object is manually designated (say by using an overlay wireframe) and its 3D position and orientation T_{vis}^{obj} is computed by vision using (*) calibration correction. The EE then is positioned (autonomously) at $T_{vis}^{obj}A$ which is probably 2 – 3 cm away from the desired position $T_{abs}^{obj}A$. Then the position of the EE is verified by the Vision as T_{vis}^{EE} and eq. (16) is used. After the EE is accurately positioned, grappling of parts mating operation can be done by using compline motion, etc.

The preliminary experiments on the JPL Telerobotics Testbed have shown that the described method allows reduction of the EE positioning error from about 2 cm to 1 mm with no noticeable orientation errors.

The most serious problem now is to design a "Vision – friendly" EE to facilitate fast and accurate EE verification by the Vision System.

In the future research, if vision bandwidth allows, a Vision based motion can be implemented, by substituting a single approach point A by a sequence $[A_1, A_2, ..., A_n]$ and by vision assist hopping between A_i.

V. Summary

The Vision Based Relative Motion together with teleoperation may work as a connecting link between autonomous free motions and specific task oriented macros based on the complaint motion, proximity sensors, and objects CAD models. A right combination of these technologies will allow us to perform sophisticated assembly and servicing operations in an unstructured world by using independently calibrated systems and a limited number of robotics primitives.

A work in this direction is continuing at the JPL Telerobot Testbed.

Acknowledgment.

This work was done at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the NASA.

Literature

[Kup88] Kuperstein M., "Implementation of an Adaptive Visually-Guided Neural Controller for Single Postures". *Proc. ACC,* June 15-17, 1988, Atlanta, Ga., p. 2282.

[Stone] Stone H.W., *Kinematical Modeling, Identification, and Control of Robot Manipulators,* Klewer Academic, Boston, 1987.

[Paul] Paul R., *Robot Manipulators: Mathematics, Programming, and Control,* MIT Press, Cambridge, Mass., 1981.

[Hayati] Hayati S. and Mirmirani M., "Improving the Absolute Positioning Accuracy of Robot Manipulators", *Journal of Robotic Systems,* **Vol. II, No. IV, 1985.**

[Matij] J. Matijevic, W. Zimmerman, S. Dolinsky, "The NASA–OAST Telerobot Testbed Architecture",

Proc. NASA Conf. on Space Telerobotics, Pasadena, Jan. 31 – Feb. 2 , 1989.

[Kan 1] E. Kan, et al.,"The JPL Telerobot Operator Control Station : Part I – Hardware". *Proc. NASA Conf. on Space Telerobotics.* Pasadena, Jan. 31 – Feb. 2 , 1989.

[Kan 2] E.Kan, et al.,"The JPL Telerobot Operator Control Station : Part II – Software". *Proc. NASA Conf. on Space Telerobotics.* Pasadena, Jan. 31 – Feb. 2 , 1989.

[Stone–2] H.Stone, et al.,"Experiences with the JPL Telerobot Testbed : Issues and Insights." *Proc. NASA Conf. on Space Telerobotics.* Pasadena, Jan. 31 – Feb. 2 , 1989.

[Gennery] Gennery et al., "Sensing and Perception Research for Space Telerobotics at JPL." *IEEE Proc. on ICRA*, March 1987, Releigh, NC.

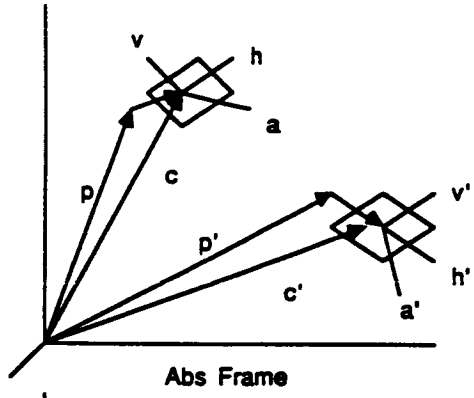


Fig. 1

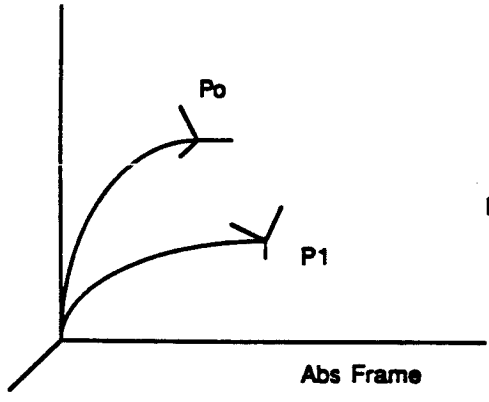


Fig. 2

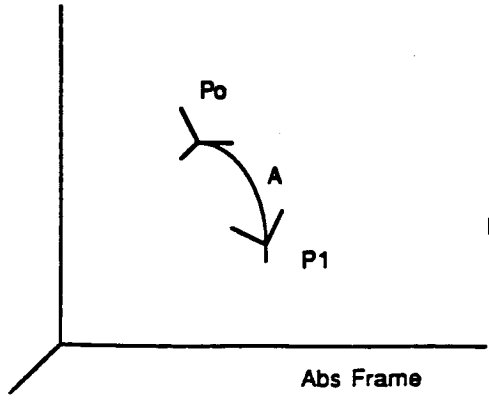


Fig. 3

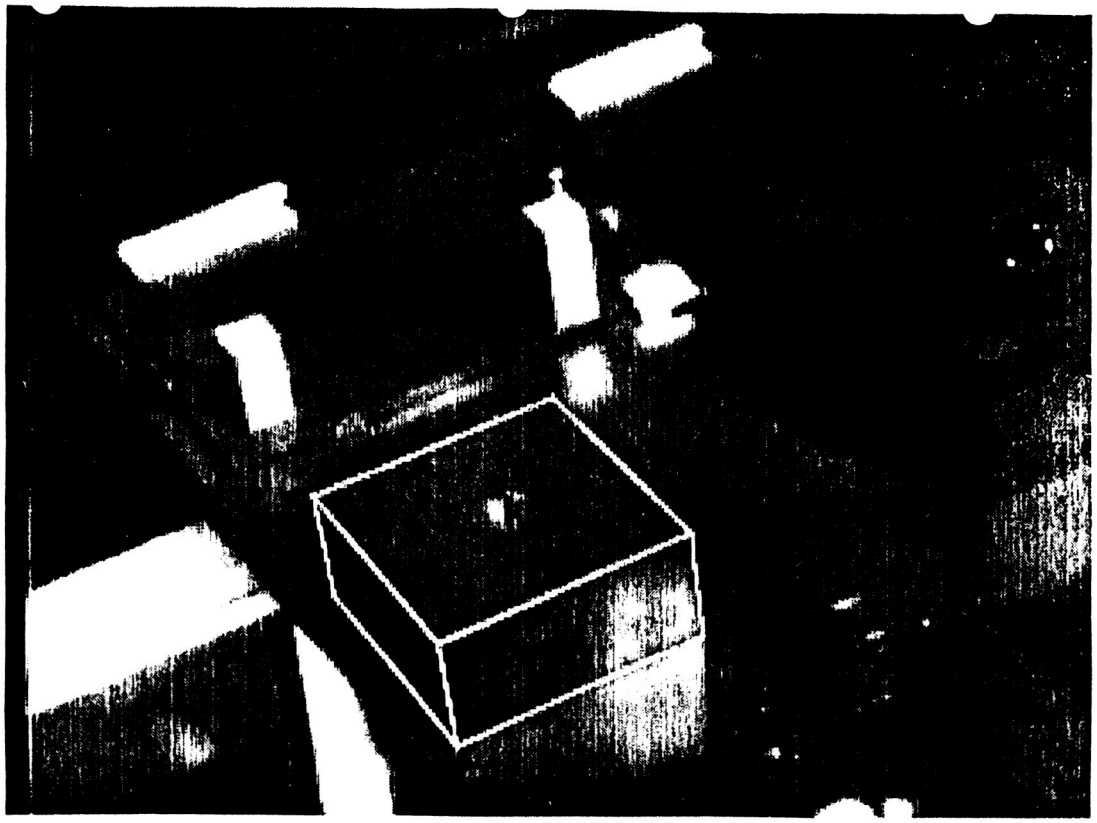


Figure 4

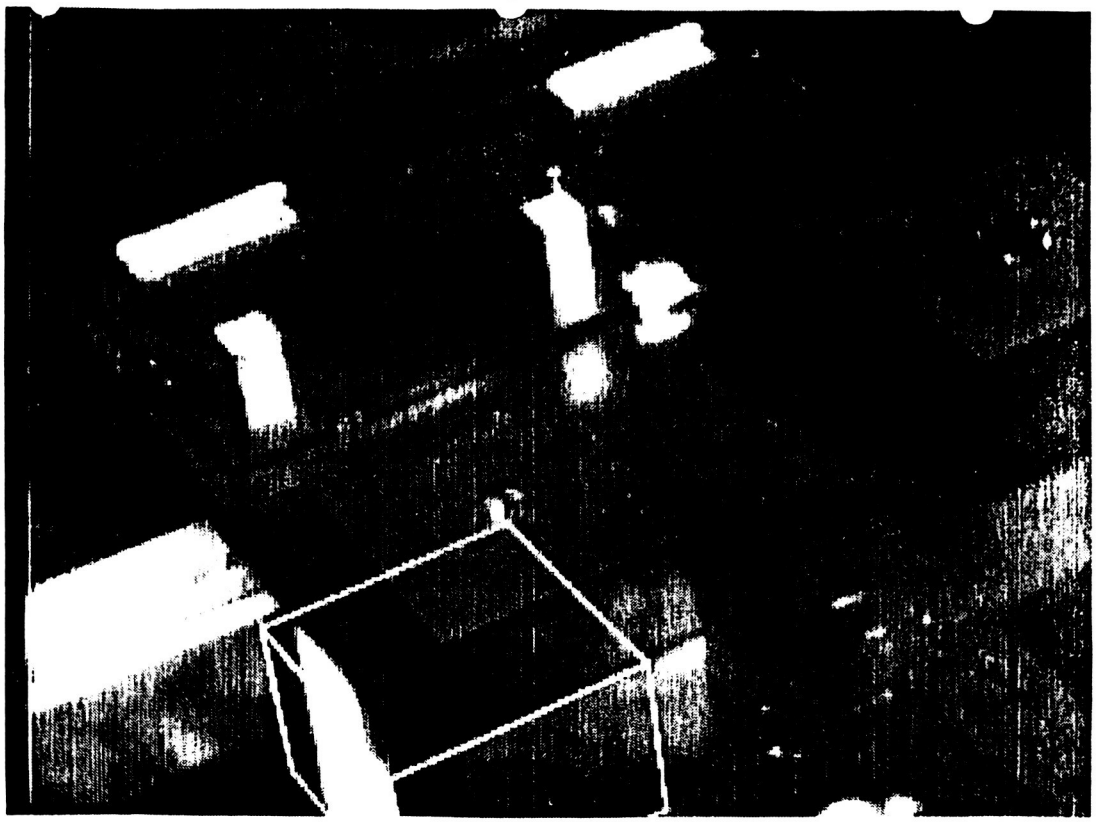


Figure 5

TELEROBOTS 2

TELEROBOTIC WORKSTATION DESIGN AID

K. Corker, E. Hudlicka, D. Young, N. Cramer

BBN System & Technologies Corporation
Cambridge, Massachusetts 02019

1.0 Introduction

Telerobot systems are being developed to support a number of space mission applications. In low earth orbit, telerobots and teleoperated manipulators will be used in shuttle operations and space station construction/maintenance. Free flying telerobotic service vehicles will be used at low and geosynchronous orbital operations. Rovers and autonomous vehicles will be equipped with telerobotic devices in planetary exploration.

In all of these systems human operators will interact with the robot system at varied levels during the scheduled operations. The human operators may be in either orbital or ground-based control systems.

In order to assure integrated system development and maximum utility across these systems, designers must be sensitive to the constraints and capabilities that the human brings to system operation and must be assisted in applying these "Human Factors" to system development. The simulation and analysis system which is the topic of this paper is intended to serve the needs of system analysis/designers as an integrated workstation in support of telerobotic design.

1.1. Design and Development in Space Telerobotic Systems

The design environment in space telerobotic systems imposes several unique constraints on the development of a design aid. This is especially true for the inclusion of human operators in system control. The telerobot system serves as an extension of human perceptual, cognitive, and manipulative abilities to a remote site of operation. The design process, then, should be responsive to human interface requirements. However, there are a number of constraints in the telerobot design process that make inclusion of operator-focussed concerns difficult to implement. Taking the development of the Telerobot Testbed at the Jet Propulsion Laboratory as characteristic of the difficulties in this large-scale human/system design, we find the following.

This work has been supported on a cooperative basis by Jet Propulsion Laboratory and NASA-Ames Research Center Contract #200-216-55

(1) Multiple Subsystems: Independently developed, but interactive in operation the telerobot subsystems (AI Planner, run-time controller, manipulator control module, sensing and perception, executive monitor) offer a challenging amount of information to a human operator/monitor. All of this information is to be channelled through the operator control station. It is this control station design that is the focus of our design aiding tool.

(2) Variable Development Cycles: Given the independent development process, the level to which subsystem function can be specified varies among the modules. This means the information provided to and action required of the operator varies in its specificity

(3) Evolving Mission Requirements: As decisions are made as to the extent of telerobot servicing required, or the level of operator intervention in that operation, the tasks and performance criteria for system operation change. The boundaries of the operator tasks profile are a moving target for interface design to capture.

(4) Undetermined Manning and Training Requirements: Because of the evolving nature of the mission and system requirements the number of operation required and whatever special training then may require is not yet specified.

In order to deal with such a fluid design environment, an aiding tool must provide the designer/analyst with a flexible, modular and modifiable system that takes into account the nature of the human operations, the nature of the operator interface, description of the mission, the constraints and capabilities of the equipment to perform the operations and some performance assessment process to judge the relative value of one design versus another.

1.2 Issues Evaluated In Space Telerobotic Design

Before describing our evaluation tool, we will briefly mention the particular issues we are attempting to resolve in relation to the JPL Telerobot Testbed.

- Control Station Layout/Design: The Operator Control Station (OCS) is the single point of interface between the operator(s) and the telerobot control subsystems, the robot arms, the satellite and the sensing and perception system. The operator will use the OCS for monitoring, control, and diagnosis of mission progress. Our design aid is intended to assist in determining: What information is needed to support the operator, what control mechanisms should be provided, how should this information be provided, and what is the optimal configuration of the OCS for operator use.

- Automatic vs. Manual Telerobotic Control: Given that the telerobot system has the capacity for control by computer or transmission of human control commands to the robot arms, there is an issue of allocation of control authority between the human and the AI Planner (AIP) subsystem. The design aid will allow system designer to explore control strategies by making assignments of varied control responsibilities for parts of the mission and examining the results of those strategies on performance and operator load. Control can be designated to reside with the operator, the AIP, or to be traded between them on portions of the task, or to be shared in task cooperative control.

- Management of the Telerobotic Systems: The coordination and diagnosis of subsystem operation is a critical task in the Telerobot Testbed. The design aid will provide the ability to allow the operator to view varied messages and message traffic among subsystems. Designers can induce simulated errors and explore diagnostic and error recovery strategies.

- Manning and Training Issues: The operational requirements for the telerobot system do not yet specify one or two operators, or suggest the operational affects of earth-based versus orbital operation. The design aid will support investigation of performance with one or two operators and allow for exploration of the impact of time delay or bandwidth limitations on feedback or control information.

2.0 Design Aid

We have taken the approach of providing telerobot designer and analysts with a simulation tool. The basic architecture of that tool is illustrated in Figure 1. The architecture is illustrative of the modular nature of the overall aiding system implementation. We will describe that implementation, and then detail the function of the major components as applied to the Telerobot Testbed.

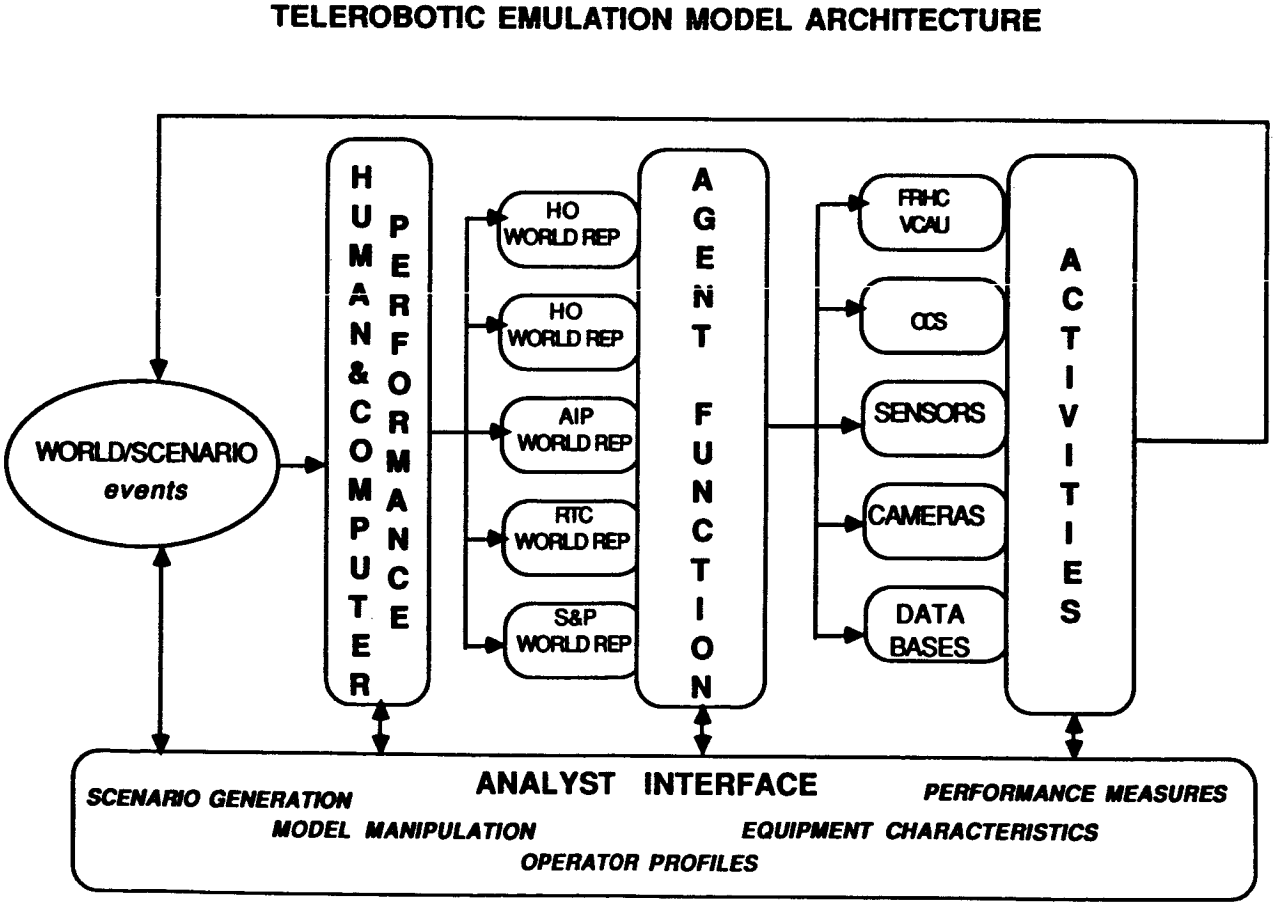


Figure 1

2.1 System Implementation

The system is implemented in Zeta Lisp in the Symbolics LISP machine environment. It uses the FLAVORS™ object system. As a Common Lisp specification for object-oriented programming develops, the system will be converted to Common Lisp and its associated object language. Common Lisp is expected to become the shared language of the artificial intelligence and LISP communities; this conversion will make this system compatible with a wide range of other systems.

The aiding system has three components:

- a simulation driver,
- libraries of object descriptions,
- and a library of input/output utilities,

The simulation driver allows a user to run the simulation at fixed (and user-definable) increments of time, or by letting the occurrence of events drive the clock, providing a discrete time, or discrete event simulation. The libraries of object descriptions include control station configuration and function, robot arms, sensors, and controls, human operator's control strategies, perceptual and motor characteristics, decision and diagnostic procedures, the task environment and tools, and a variety of other supporting object types. These will be detailed below. The input and output utilities include the ability to display the position and status of the mobile objects in the system, to display the procedural hierarchy of activities, to display inter subsystem communication, and to display the level and type of loading on the operator.

2.2 Functional Modules

Scenario: In order to give the designer a tool to exercise the telerobot system and operator models in a variety of operations, we have created a module of object code that contains operational procedures. These procedures are examinable and modifitable using screen-based tools.

We have selected a portion of the telerobotic demonstration to exercise our simulation tools. Beginning with the "flat" sequence of planned activities and communication among the system models that was used to encode the telerobot scenario, we have constructed a procedural hierarchy of goal-oriented activities organized from the point of view of the "cognitive agents" in the simulation, i.e., the AI planner and the human operator.

The scenario operation is represented by a semi-lattice of goals, subgoals, and activities and procedures for meeting those goals. (Figure 2). This representation provides an abstraction of the scenario activities. The abstraction provides the designer/analyst with an overall view of allowable and selected paths to the desired sequences of activities to accomplish the demonstration tasks. It also provides a convenient tool for alternative plan construction and task-level diagnostic activity. The procedural representation, the logical and procedural constraints that determine allowable actions will be made available as a mouse-sensitive object display. This display will show the current sequences of simulator action. When the activity nodes are selected a more detailed description of that activity will be provided. The details of an activity such as its duration, or preconditions, or termination requirements, tolerance, or agent required to perform that activity are all modifiable by the designer.

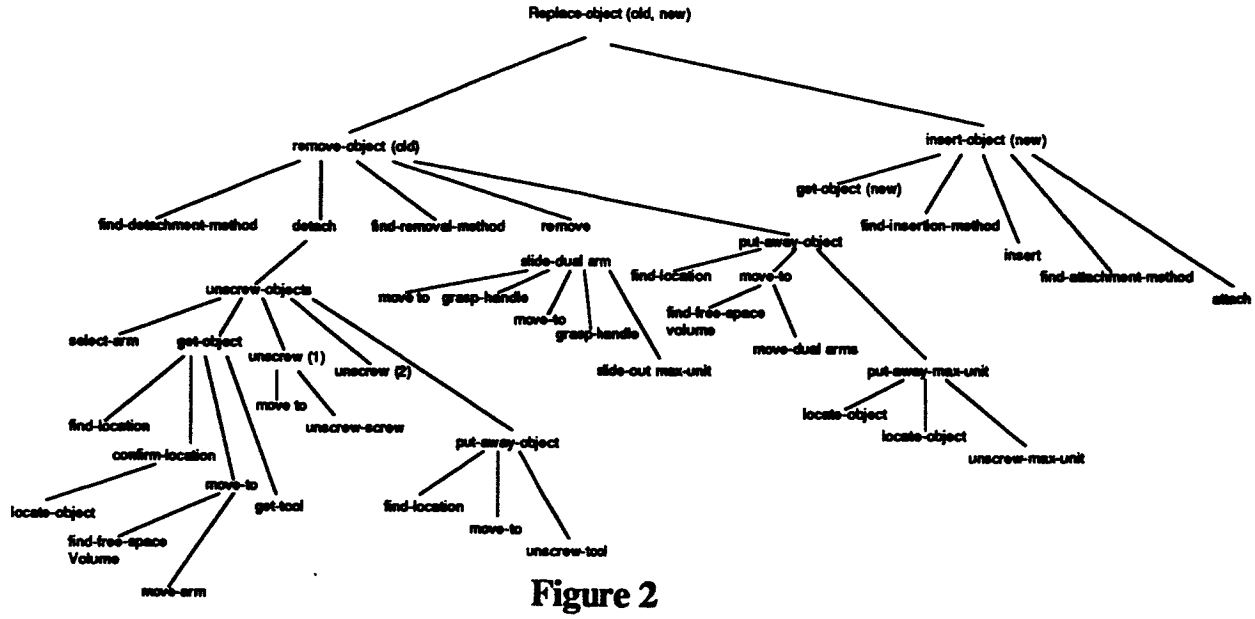


Figure 2

Human Operator Models: The framework in which the individual models of human performance are embedded is the "operator object". The operator object interacts with the telerobotic demonstration through perceptual processes and activities. The operator has an individually defined "updatable world representation". This world representation is a description of the world as the operator knows it. It contains rules for decision, and an awareness of external events as they are passed through the operator's sensory systems. Each of the slots of the operator object's world state contain information as to the source of the information in the slot. For example, an alert state will contain a slot identifying the source of the information about that alert, e.g., "flashing warning, position X, Y, on the control station." The world representation in the object is a decaying store of information. World state changes are entered into the representation with time signatures and priorities. The frequency with which state changes occur and their relative priorities, will determine the length of time that state information remains available for decisions and rule applicable. The world representation also contains a queue of action that should be taken at some time in the future, a "goal queue". The operator interacts with the task-world through sensory systems.

These sensory processes are modeled by our system. We have implemented a visual model.

Visual Scanning: Each of the activities in the simulation script has an attribute which identifies what equipment (and what sequence of interaction) it requires. The operator(s) of the telerobotic system must constantly "scan" their equipment and displays to keep high resolution (foveal) information updated. In order to account for the time and movement required to find and fix target data in the telerobotic tasks, we are implementing a model of visual scanning in two stages.

The first "active gaze" represents the focused and directed movement from the current point of regard to a target point. The action is characteristic of actions in which the to be attended object is in a known position. The motion is a straight line from the present position to the target (though there may be a contribution through head movement, we will not consider that at this time.) The operator is assumed to be 18 in. from the center of the OCS. The velocity of motion is 100 degrees per second and the foveal fixation area is .5 degrees. There is a 200 msec. pause between eye motions (i.e., saccades).

The second type of gaze is a monitoring or search pattern. The saccades in in such a search pattern typically last for 50 msec and cover about 10 degrees. Again there is a 200 msec pause between movements. The effective radius of a fixation in this scan is about 14 degrees around the center of fixation.

In order for an operator to "notice" any information or to make judgments to guide teleoperation, the operator object must have scanned the relevant source of visual information. If the operator's visual attention is focused on a particular display or control he/she will not respond to other information until it is viewed.

This model provides information about visual latencies and the positioning of displays on the OCS. We hope to implement more sophisticated visual model to include depth perception and visual acuity issues for display design. We have also implemented a decision-making process.

Decision Making and Diagnosis: The operator object's updatable world representation contains object descriptions of those actions and events on which decisions need to be made. Using a rule-based decision framework, the operator object applies rules to the world representation to determine what action is required. If the rules are insufficient to make a decision (and an activity selection is required) then the operator object will look at the full procedure hierarchy to select alternative and appropriate actions to follow. The approximate time required to make a decision or to pursue a diagnostic path is calculated and that time is added to the simulation performance time.

The idea of the human operator as a limited capacity channel in complex system operation is central to our model. Designers should be aware of the attentional and workload implication of a given design. In order to give some insight into the loading on the operator of a given operation we have implemented a performance capacity model for the operator.

Task and Operator Loads: As previously described, tasks are represented in a hierarchy of procedure. At each level of that hierarchy an assessment is made of the load associated with that activities performance. These assessments are currently estimates, but future activity descriptions will have estimates provided by expert's subjective opinion and represented on a scale from 0-7 with 7 indicating complete or full load. These task/activity loads are further refined by being divided according to resources that an operator can bring to hear to perform the task. We have partitioned those resources to be visual, auditory (which includes both speech and hearing), cognitive and psychomotor (VACP). In addition, we are implementing a moderating or gain function, which we denote as attention, on the use of those resources. The process by which the task demands are met by the operator resources is mediated by attention.

Efficient matching of resources to demands, i.e., resources applied equal to and for a duration commensurate with demands, can be expected of a well-trained and unstressed operator. In appropriate allocation of resources is the result of lack of training or a change in the attention allocation function.

Agent Functions: Agent functions stand as links between models are going to carryout. The agent functions to free the designer from the assumption that a particular activity is to be performed by one or more human operators or by an intelligent machine. For example, communication (as an agent function) knows that in order to communicate there is a source and a destination for information and a message to be passed. If the communication is among submodules of the telerobot system then one type of message protocol and one type or resource is involved, i.e., a network transactions protocol, and bus or fibernet links. If human operators are communicating then auditory models are invoked. Human communication with the system takes place through the OCS. Figure 3 illustrates the role of agents linking operators to activities.

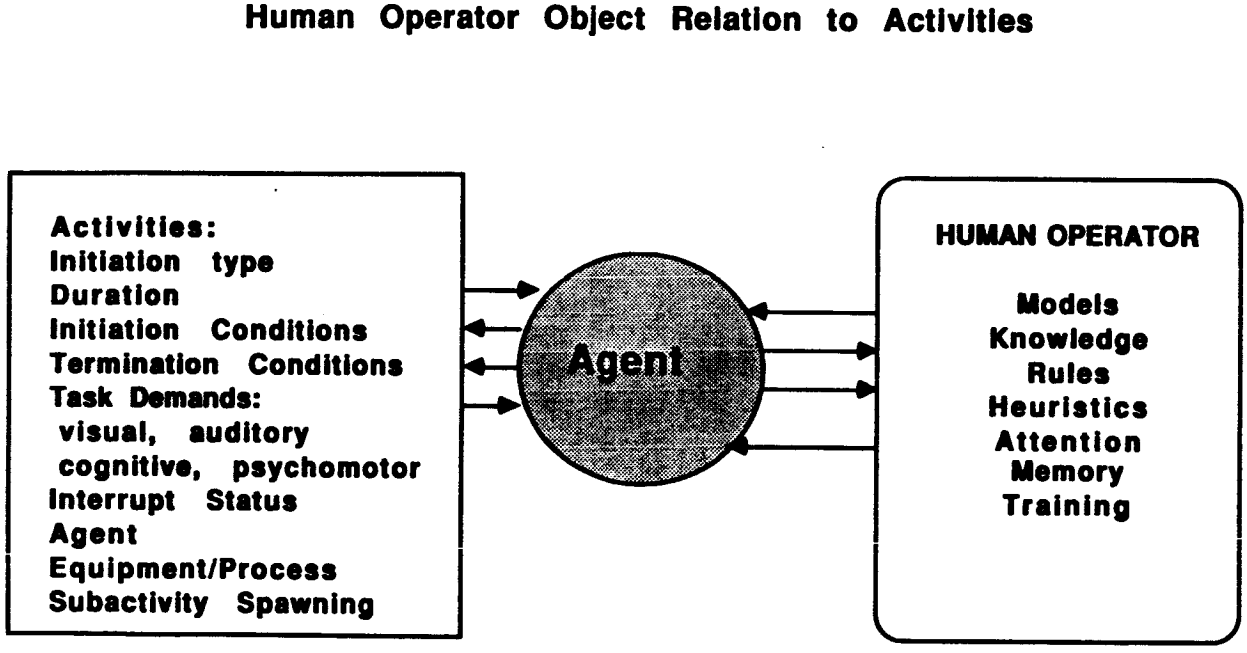


Figure 3

Equipment: We have focussed on the OCS design in our simulation. Figure 4.

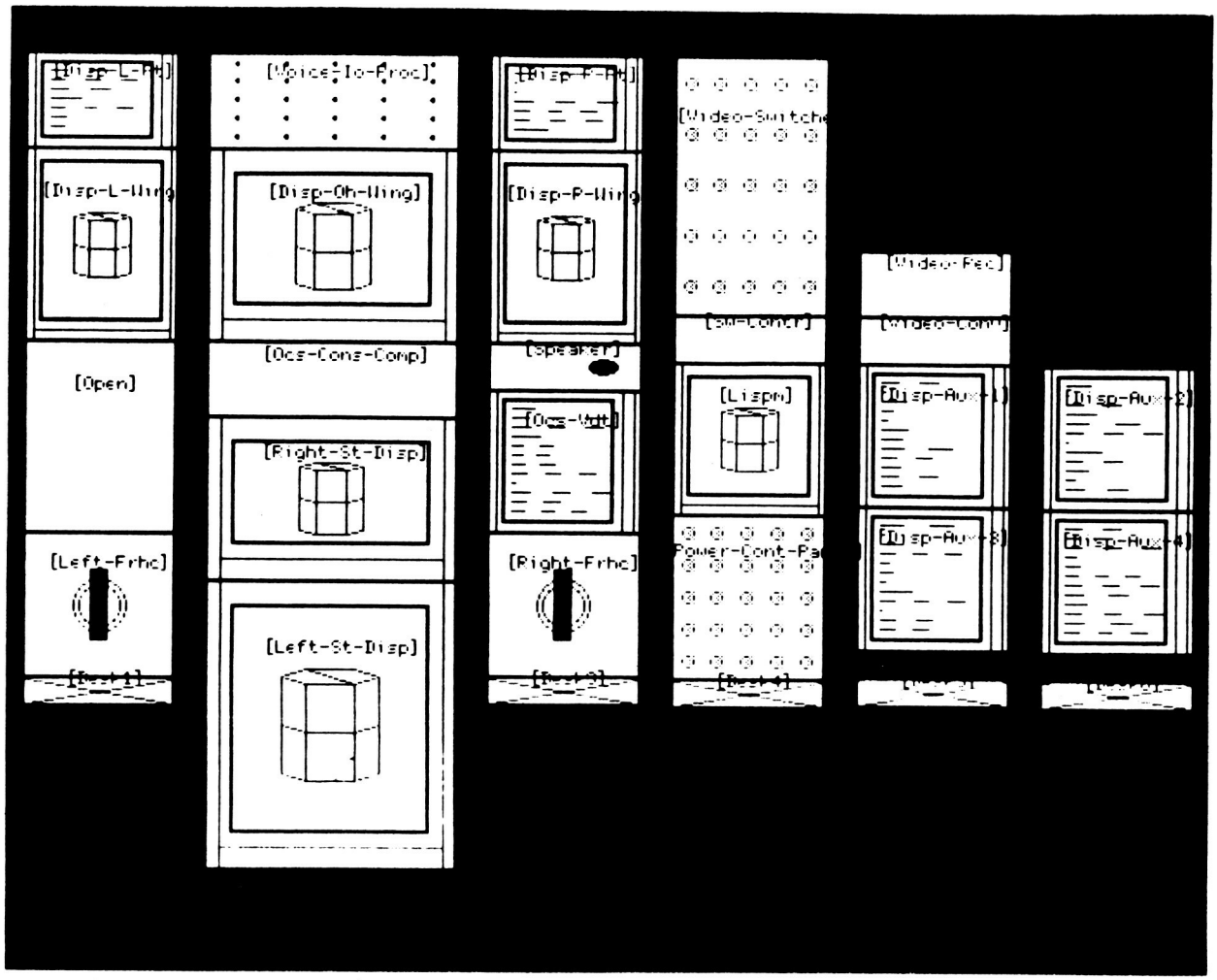


Figure 4

Illustrates an object-based control panel. The switches, controls and displays are objects whose function moves if the placement of the device is changed. All of the control and display functions are "Mouse Sensitive" and can be selected and moved to reconfigure the layout of the control station. The switches and controls can be made "Active" in the mousing them initiates the appropriate function in the software simulation. The panel is, therefore, a reconfigurable display the control. The information or control that is relevant at a particular moment in the simulation is also highlighted as the sequence of mission operations occur. Designers can reconfigure the control

station, assume different types of display, or levels of automation and then run the simulation with the candidate configuration. The visual model and workman will respond to a given configuration with a unique profile of operator activity. We also provide a view of the ongoing robotic task. Figure 5. We also represent communication among the subsystem components. Figure 6. This display is screen assessable so that the design can track decision and control among components.

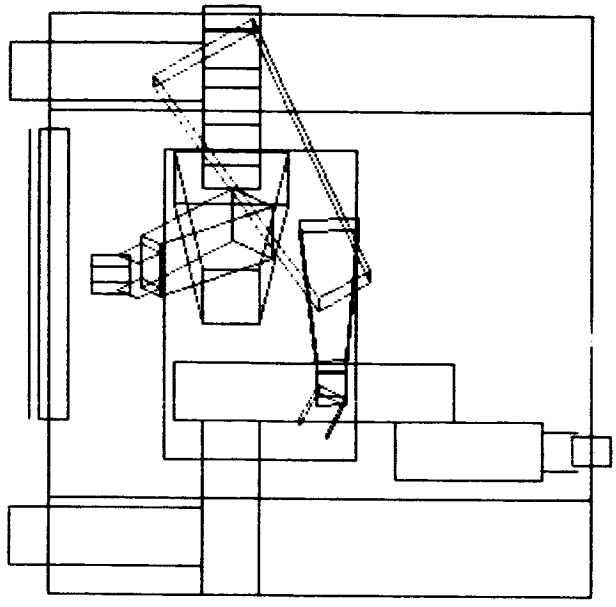


Figure 5

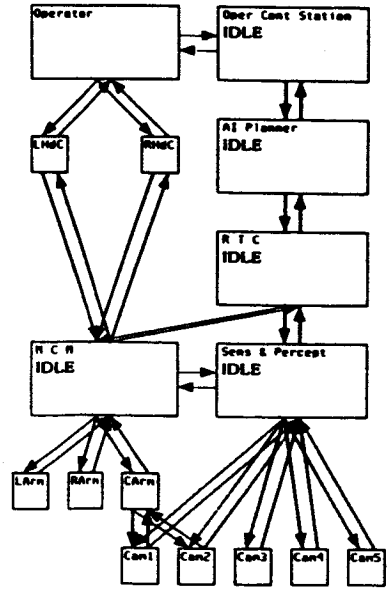


Figure 6

Activities: Each activity has operations that describe its function at each time increment ("tick") of the simulation, when the activity is initialized, and when the activity is terminated. For example, for activities with sub-activities, the tick operation is responsible for starting a sub-activity, waiting until it has finished executing, and then starting the next sub-activity. When an activity is created, its tick operator performs the instructions described in the activity's initialization procedures variable, and when an activity is finished executing it's termination procedures are performed.

FLAVORS ™ can be defined by combining several other FLAVORS™ and adding new operations or state variables. In this way, base activity types are created to support activities which can perform their sub-activities:

- either sequentially or in parallel,
- allowing interruption or not,
- with a fixed or variable execution time,
- allowing further subactivity spawning or not.

By making instances of activity FLAVORS™ and assigning them to the appropriate active-objects and agent, the crew-member/agent/can "tick" their activities and take further actions based on the state(s) of their current actions.

Activities themselves are organized in a hierarchic structure. An activity has a parent that spawned it, a list of children that it has spawned, a procedure to carry out when it receives an act message, and a set of procedures to carry out when it is created, terminated, interrupted, suspended, and/or aborted.

Activities can be performed by one or several agents. The evaluation system requires no assumption as to the particular technology used in performing an action, nor does it require rigid specification of the results of action. The message-passing protocol establishes a modular discipline on the development of the simulation environment. This allows, for example, an investigation of performance assuming an automatic control for a task compared with the same task performed in a teleoperated mode. The changes in system response are propagated through the simulation/evaluation as a function of the message passing protocols associated with the object-oriented code. The demonstration assumes that an specific "active object" will normally perform a given activity. An active object is defined as an object that has a list of activities that it is carrying out. When an agent is active (i.e., when it has received an appropriate message) it activates each of its activities and they proceed to carry out their function. This type of object, the active object agent" is a component of many other types of objects, including telerobots, AI Planner, human operators and etc.

Performance Measurement: In order to be use to analysts in evaluating design alternatives, some measure of performance must be provided by the system. Figure 7. The performance and activity tracking window for a human operator in a telerobotic scenario. The histogram represents work load for the human operator at each instance of operation time along visual, auditory, cognitive and psychomotor dimensions text to the right is the currently active task hierarchy for each of the significant subsystems of the telerobotic demonstration system. All of the displays are objects that are "Mouse Sensitive" and interactive, and more detailed information is available for any of the activities. In addition in activity file of actions-performed is maintained and is available for statistical manipulations.

3.0 Discussion

The object-oriented, model-based simulation described provides the designer of telerobotic systems and operations a tool to investigate the feasibility and performance impact of various mixes of human operator and autonomous control in space telerobotics. The system produces efficient and effective methods for early identification of procedural bottlenecks and control conflict. Iterative refinement of the design of control architecture can focus the designer's attention and resources on particularly promising or troublesome interactions. Once identified these critical interaction points can be explored in part-task simulation. Additionally, the system provides a preliminary prediction of operator performance given a particular control station configuration. Issues of control station usability can then be addressed prior to commitment to a prototype configuration. Finally, the system design tool serves as a repository of data as performance and prototyping experiments are performed on telerobotic component modules, and the design is incrementally decided upon.

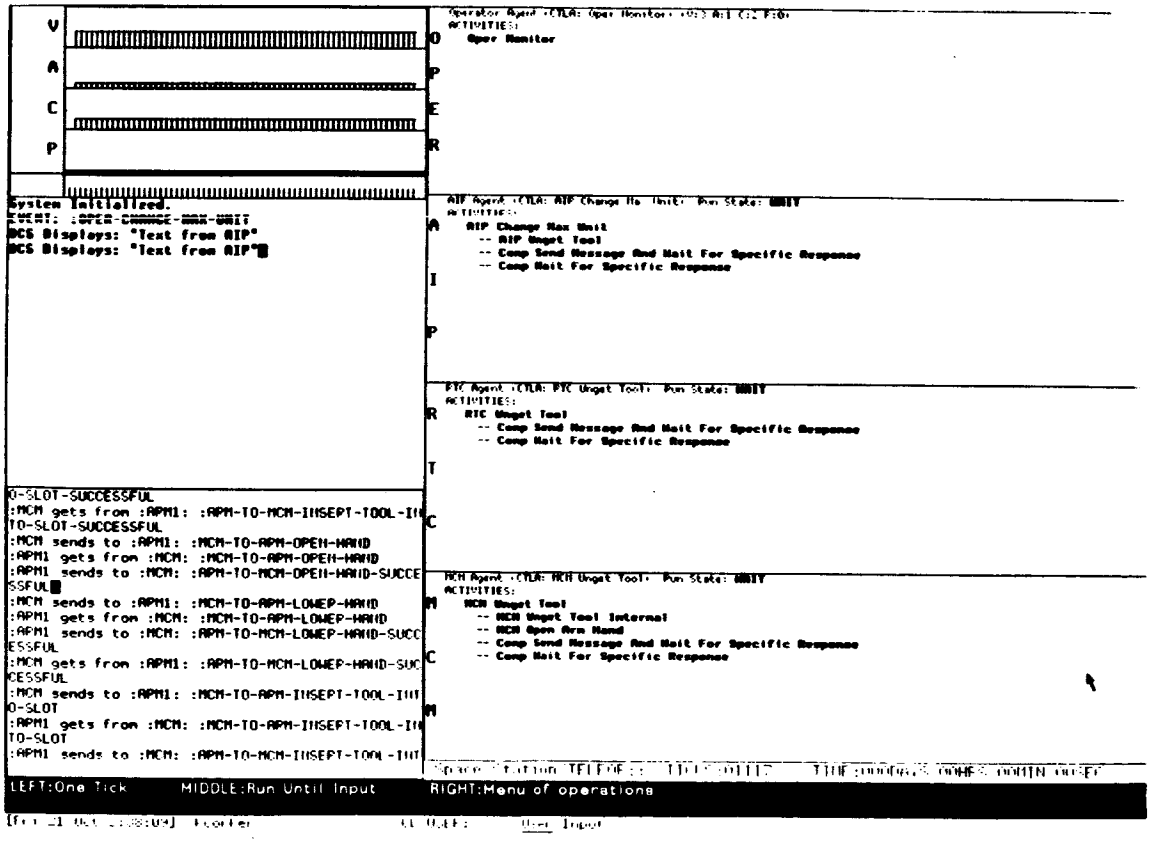


Figure 7

SPACE ROBOTIC SYSTEM FOR PROXIMITY OPERATIONS

P.G. MAGNANI, M. COLOMBA

Tecnospazio S.p.A.
Via Mercantesse,3 - 20021 Baranzate di Bollate
MILANO (ITALY)

Abstract

Space stations will be built, maintened and upgraded (servicing) in space over a period of several years. Key to an efficient accomplishment of servicing operations is the development of a scenario where the presence of man in space is well integrated with the capability of teleoperated and automatic robot system outside the stations. In this context, Tecnospazio is performing, on behalf of the Italian Space Agency (ASI), a feasibility study on a space robotic vehicle (SPIDER) capable of inspection and repair activities around and on Space Platforms.
The paper illustrates the results up to now obtained and, in particular, will focus on mission requirements, trajectory sequences, propulsion S/S features and manipulative kit characteristics relevant to the MTFF-RM proximity servicing mission (Robotic Mission). Nevertheless, the type of vehicle here considered can operate in different scenarios, e.g. in Space Station close proximity, provided that the overall energy budget is maintened.

1. Introduction

A generic on orbit servicing mission is schematically represented in **Figure 1-1.** Such a mission is formed by two main portions, functionally separated:

- the Logistic Mission
- the Robotic Mission

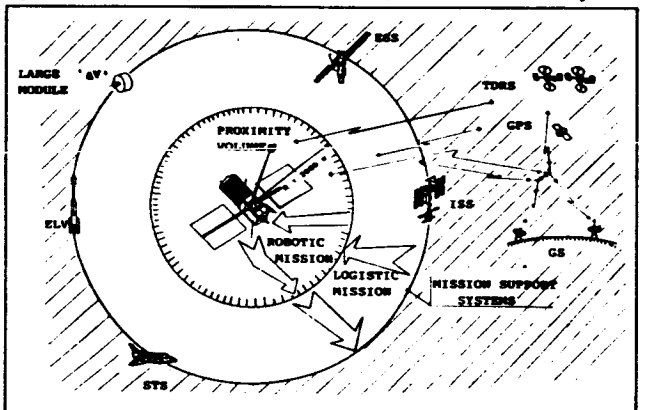


Fig. 1-1: On Orbit Servicing Mission

The Serviced System (the MTFF-RM) is considered included in a "Proximity Volume" (for example a 2000 m. diameter sphere) inside which the cold gas propulsion is advisable. The Mission Support Systems are constituted by large platforms and/or structures allowing proper interfaces with the Telerobot.
The Logistic Mission takes care of the Telerobot phisical transfer between the Mission Support Systems and the Proximity Volume. The transfer, normally performed by large ΔV modules, is controlled through high level requirements such as: mission synchronism, orbital parameters, energy, etc.
The Robotic Mission is intended to be constituted of all the activities to be performed by the Telerobot inside the "Proximity Volume".
Tecnospazio on going activity is related to a feasibility study of a Telerobot capable to perform the Robotic Mission. The activities performed are in line with general "feasibility aspects" rather than optimization or detailed technological considerations except for "technological risks".

2. Robotic Mission Requirements

The starting mission condition (see **Figure 2-1**) is assumed to be the following:

- the Telerobot is docked to a large ΔV transfer module and parked (on the surface of the proximity volume) at x = +2000 (m) along the + VBAR.

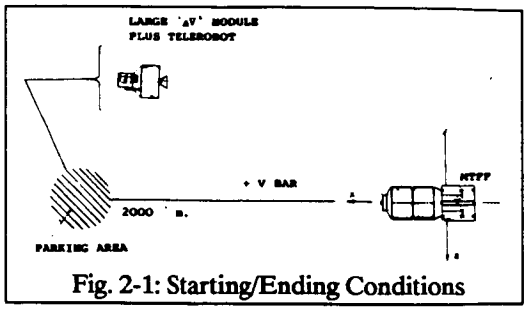


Fig. 2-1: Starting/Ending Conditions

The ending mission condition is assumed to be the same as the starting one.
The activities to be performed during the Robotic Mission and which constitute the Telerobot requirements, are shown in **Table 2-1.**
It must be pointed out that the identified requirements are derived from a MTFF configuration still provisional and with the specific servicing requirements not yet fully defined. Therefore, some of the requirements are necessarily in a qualitative form though allowing a range of possibilities.
Based on activities already performed, and anticipating some outputs, the candidate Telerobot concept, presently under investigation, is represented in **Figure 2-2.**

It is formed by the following main subsystems (S/Ss):

- cold gas propulsion
- manipulative (arms + rigidizer)
- storage area
- sensors
- electronics (computing, processing, Man Machine Interface (MMI), Telemetry, Tracking & Command (TTC), etc)
- mechanical body

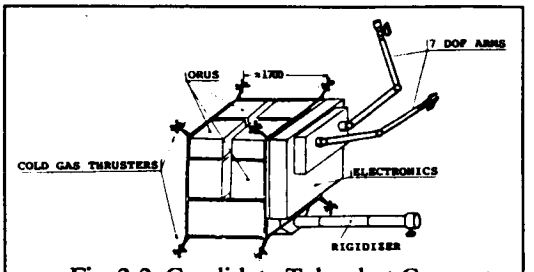


Fig. 2-2: Candidate Telerobot Concept

The "size" of the investigated concept is maintened in a reduced envelope based on three main assumptions:

- a small and dexterous robot can include and cover the gap between the work space envelopes typical of human EVA and of the manipulators in the 10 - 15 m. class
- a small robot needs support system with "limited capabilities"
- the Italian Space Agency (ASI) is oriented in the development of this class of Telerobot (SPIDER).

Prox Navigation Requirements

- Telerobot transfer from parking zone to MTFF and return
- Telerobot to MTFF closed (1 - 20(m)) proximity motion
- Approach/Docking capability
- Attainment of safety criteria

Inspection Requirement

- In "prox" condition (100 - 2000 (m))
  - visual (definition in the order of 1 - 10 (mm)
  - I/R (thermal mapping)
- In "closed prox" condition (1 - 20 (m))
  - visual (TBD)
  - I/R (TBD)
- In "docked" condition
  - temperature (contact)
  - leak detection
  - detailed optical

Manipulative (arms + rigidizer) Requirements

- Exchange of 5 ORU (0.7 x 0.7 x 0.5 (m) 70 (Kg)) each in RM Main Body
- Opening/Closing of thermal covers in RM Main Body and Super ORU section
- Telerobot transfer between two docking ports (in continuous mechanical contact)
- Alternative possibility of exchanging 1 non standard ORU (assumed features: 0.9 x 0.9 x 1 (m). 350 (Kg)) located in the super ORU section.

Tab. 2-1: Robotic Mission Requirements

3. Propulsion S/S Evaluation

The propulsion S/S considered is a cold-gas low-thrust type capable of smooth, non polluting proximity motion.
Three types of missions (see **Figure 3.1** for a general schema) have been considered in order to derive the S/S features:

- Fly-by
- Docking
- Closed inspection.

The fly-by and docking are two basic and fully dedicated missions. The closed inspection mission is a variant that can be added to each of the previous ones. The computed ΔV requirements include:

- full thrusters misalignement effects
- margin for attitude stabilization
- margin for dispersion recovery
- safety margin.

Safety Criteria

The safety criteria utilized in order to derive the specific trajectory features is herebelow described, with respect to "off-type" thrusters failure (see **Figure 3-2**):

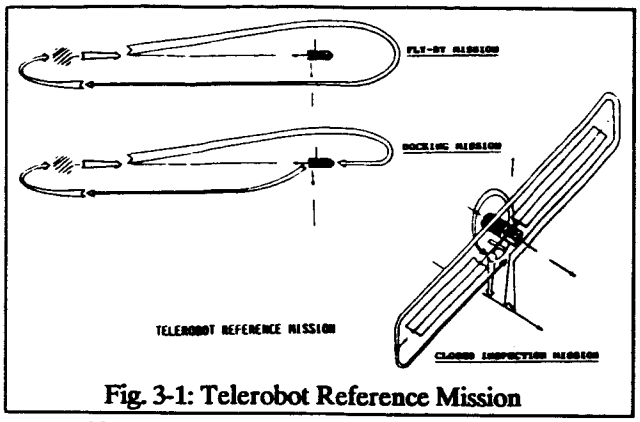


Fig. 3-1: Telerobot Reference Mission

any manoeuvre $X(\tau)$ in the Σ_2 space is considered "safe" if a complete "off-type" thrusters failure at time $\tau = t_i$ will leave the telerobot in a drift orbit that exclude the regions Σ_1 and Σ_3 or enter these regions with a speed < 0.01 m/s for any time $\tau > t_i$.
The speed of 0.01 (m/s) is chosen in order to guarantee a "non-damaging telerobot/stations collision". The free drift trajectories following an "off failure" are represented, in the next figures, by dotted lines.

Fly-by Mission

The Fly-by mission has the objective of performing MTFF external inspection in "far" conditions (90÷600 m. range). The mission is made up by three main trajectories (**Figure 3-3**) hereafter summarized:

- Trajectory A: it consists of two impulsive and one continuous steps
- Trajectory B: it consists of four impulsive steps and two coasting phases
- Trajectory C: it consists of three impulsive and one continuous steps and one coasting phase.

The requirements for this mission result in the following:

$\Delta V = 2.78$ (m/s)
$T_m = 50{,}619$ (s)

where T_m is the manoeuvre time. In strict proximity the manoeuvres are performed at $V_m = 0.01$ (m/s) for safety reasons.

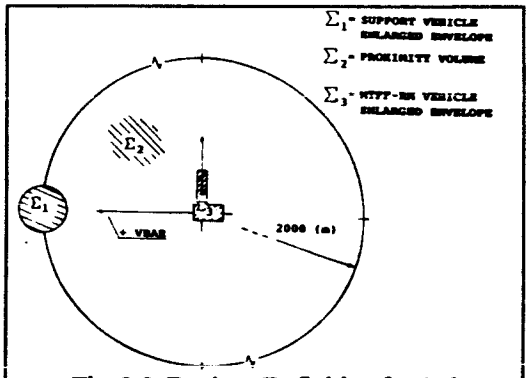


Fig. 3-2: Regions Definition for Safety

Docking Mission

The Docking mission has the objective of performing MTFF-RM external servicing in docked conditions.
The mission is made-up by four main trajectories (**Figure 3-4**) hereafter summarized:

- Trajectory A: it consists of two impulsive and one continuous steps
- Trajectory B: it consists of eight impulsive and one continuous steps and three coasting phases
- Trajectory C: it consist of four impulsive and one continuous steps and two coasting phases.
- Trajectory D: it consists of three impulsive and one continuous steps and one coasting phase.

The requirements for this mission result in the following:

$\Delta V = 5.86$ (m/s)
$T_m = 79{,}852$ (s).

The time required for the manipulator tasks has to be added to the computed time manoeuvre Tm.
The continuous strict proximity manoeuvres are performed at $V_m = 0.01$ (m/s) for safety reasons.

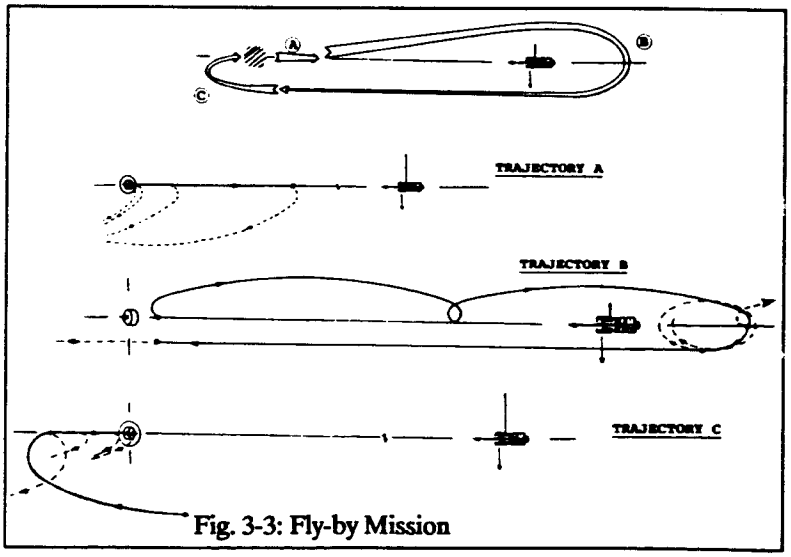


Fig. 3-3: Fly-by Mission

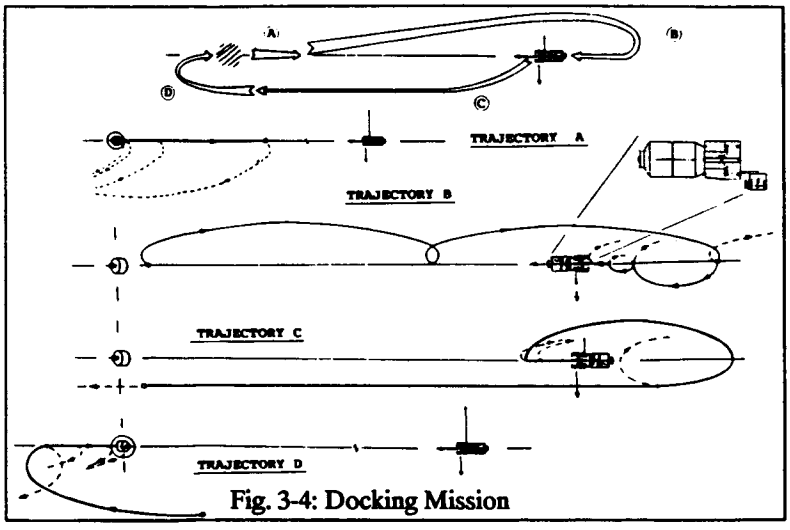


Fig. 3-4: Docking Mission

Closed Inspection Mission

The Closed Inspection Mission has the objective of performing MTFF closed inspection (few meters) and is made-up by a number of trajectories (**Figure 3-5**) including impulsive and continuous steps. This is a kind of Δ mission and could be added to both the Fly-by and Docking ones: the injection point is in the return phase at the coordinates X = 0 (m); Z = 90 ÷ 100 (m). This is a non-safe mission unless the manoeuver speed is kept below 0.01 (m/s). Unfortunately, the ΔV requirement is a function of the manoeuver speed and presents a minimum of ΔV min = 7.45 (m/s) for V_m = 0.05 (m/s).

Maintaining the manoeuvre speed within the safety level, the mission requirements are the following:

V_m = 0.01 (m/s)
ΔV = 16.6 (m/s)
T_m = 39,142 (s).

Reference Overall Mission

The evaluations summarized in the previous points allow the definition of the following reference mission:

Reference mission = Docking mission + Closed Inspection Mission

ΔV_{Ref} = 22.4 (m/s)
T_{mRef} = 119,100 + TBD (s)

where TBD refers to the time required for operation in docked configuration.

Ejected and Stored Fuel Requirements

The propellent utilized is constituted of GN_2 nitrogen initially stored at 3000 (psia) pressure, while the thrusters considered for this type of application allow a specific impulse in the 65 ÷ 70 (s) range depending on the actual tank pressure.

The overall ejected fuel is then computed through the rocket equation. The stored fuel mass is computed through a 15% increase of the ejected fuel mass to account for a residual 250 psia tank pressure. The results illustrated in **Figure 3-6** are summarized as:

M_{EF} (Ejected) = 51 (Kg.)
M_{SF} (Stored) = 58 (Kg.).

Propulsion S/S main features

The propulsion S/S is schematized in **Figure 3-7** and is fully redundant as far as thrusters, piping and valves are concerned.

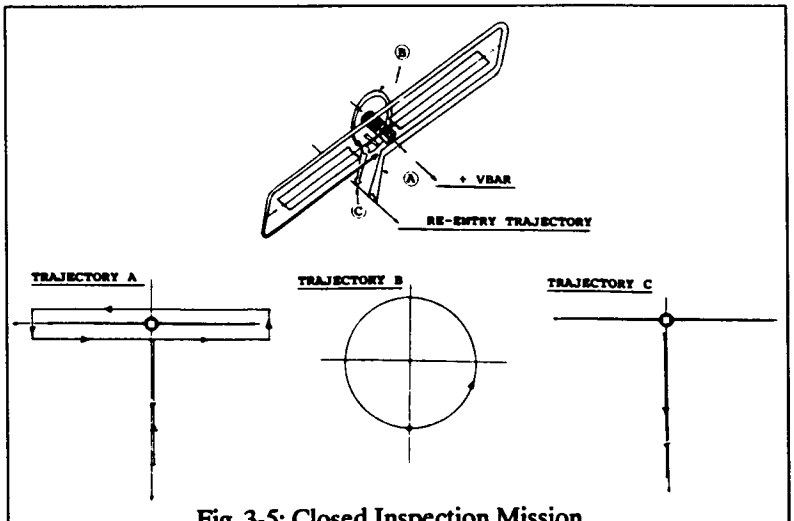


Fig. 3-5: Closed Inspection Mission

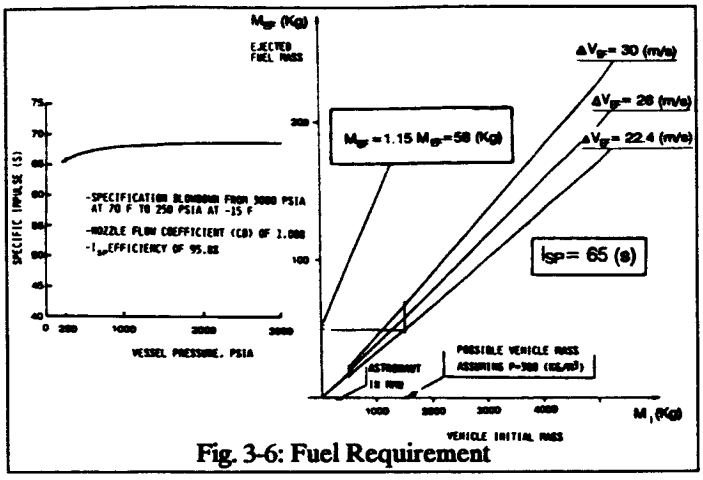


Fig. 3-6: Fuel Requirement

It consists of the following main components:

- 24 (eight triad) fixed position nitrogen (GN_2) /thrusters
- 2 tanks for nitrogen
- piping and valving connections
- Thrusters and Drive Electronics
- (Prox Nav Computer)
- (Nav Sensors).

Figure 3-8 illustrates the main component features.

Thruster triads.

Each triad module contains three identical 10 N class thrusters; each of them can be independently activated through an elctrically operated solenoid. A number of cold thrusters is already available on the market.

Nitrogen Tanks.

Two tanks, each one with 133 (liters) volume and 26 (kg.) nitrogen loading, will be utilized. They can be manufactured following a composite design. No special technological difficulties are foreseen.

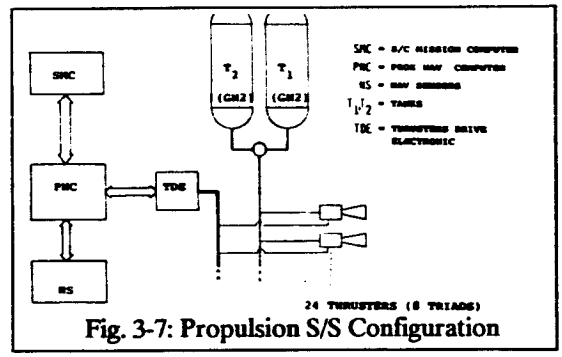


Fig. 3-7: Propulsion S/S Configuration

Piping and valving system.

The piping and valving system is duplicated for the primary and redunded thrusters and it is formed by a number of connectors, valves, regulators and pipes. No technological difficulties are foreseen.

Thrusters Drive and Electronic Unit (TDEU).

The TDEU converts the command signals, coming from the Prox Nav Computer, into currents to thruster windings and consists of three main sections: Power and Data Interface (I/F), Driver Selector, Command (CMD) generators. No technological difficulties are foreseen.

(Proximity Navigation Computer and Navigation Sensors).

These items will be considered in the prosecution of the activity.

Propulsion S/S Overall Budget.

The propulsion S/S overall budget is summarized in **Table 3-1** with respect to mass, volume, energy requirements and expected technological criticalities.

4. Manipulative S/S Evaluation

The manipulative S/S considered is formed by a bi-arm system, with two identical 7 d.o.f. arms, and a rigidizer mechanism which is a 3 d.o.f. device.
The arms are in the 2 - 3 meters class according to the general requirement of dimension reduction.
For this reason, it is necessary to identify on the serviced vehicle a number of different docking points which allow, through a combined arms/rigidizer action, the displacements of the servicing system on the surface of the serviced vehicle.

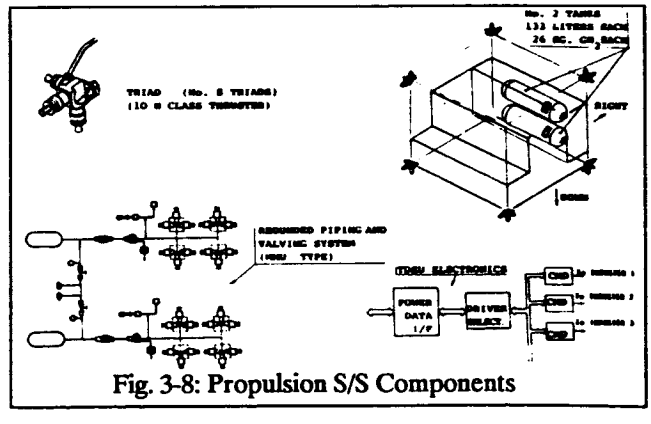


Fig. 3-8: Propulsion S/S Components

FEATURE / ITEM	MASS (Kg)	VOLUME (CM)	ENERGY (Joule)	SINGLE ITEM TECHNOLOGICAL RISKS
THRUSTER TRIADS	15	"SPECIAL LOCATION"	200,000	OFF THE SHELF
ON BOARD GN_2	58	"LOCATED INSIDE TANKS"	N.A.	NONE
TANKS	42	0.27	N.A.	NONE
PIPING AND VALVING SYSTEM	15	INCLUDED IN THE TOTAL THROUGH MARGIN	LIMITED INCLUDED IN THE TOTAL THROUGH MARGIN	NONE
TDE	5	0.008	40,000	NONE
TOTAL (Including +5% margin)	≈ 142	≈ 0.29	255,000	

Tab. 3-1: Propulsion S/S Overall Budget

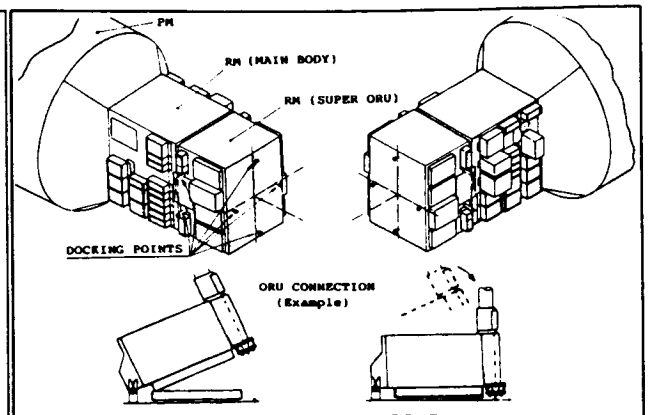


Fig. 4-1: MTFF-RM ORUs Layout

The rigidizer is basically an extendable boom (1 translational d.o.f.) with 2 additional rotational d.o.f. near the End Effector. Its length in extended position will be about 2 meters.
The manipulative S/S has then three basic goals:

- Orbital Replaceable Units (ORUs) exchange
- Displacements of servicing system around the serviced vehicle (with continuos mechanical contact)
- On orbit repair and smart manipulative actions (which can be accomplished in future evolutions).

Reference Servicing Mission and Requirements.

The development of MTFF design has pointed out the necessity of changing 8 - 10 external ORUs every year. For this purpose, 2 Hermes missions are foreseen, one every 180 days. Furthermore, an extraordinary servicing mission every three- four years is planned in order to change the SUPER-ORU: the MTFF will dock to the International Space Station (ISS) where the Remote Manipulator System (RMS) will perform the operations.
The dimensions of the standard ORUs are limited by the dimensions of Hermes, while the dimensions of the SUPER-ORU depend on those of the U.S. Space Shuttle. Actually, there are different sizes of standard ORUs, while not definitive information are available with regard to the dimensions of the non standard ORUs in the SUPER-ORU area. The dimensions of the largest size standard ORU can be so given: this ORU, with all its equipment and frames, is contained in a volume of 700x700x520 (mm) size with a mass limited to 70 Kg. The manipulative mission taken as reference for the Telerobot is related to the MTFF-RM external servicing with a capability to exchange up to 5 max size standard ORUs. The ORUs will be fixed to the RM with a standard interface (independently on their size) through a latching/delatching mechanism. The body of the servicing veichle will contain 6 interfaces: 5 standard ORUs can be fixed to it with 1 spare location necessary for the change operations.
All these requirements lead to define the servicing vehicle body as a 1.7 meter side cube.
The MTFF-RM configuration is shown in **Figure 4-1**. Comparing these dimensions with those given in the previuos paragraph, it can be assumed to have 8 docking points, 4 on the RM back, near its propulsion system, the others in the two sides of the SUPER-ORU area.
Due to the presence of thermal covers, the four docking points located in the SUPER-ORU area, can be reached only after the opening of the covers.
It is then possible to split the reference mission into 5 phases :

- 1) Opening of covers
- 2) Change of docking point (maintaining mechanical contact)
- 3) Standard ORUs exchange
- 4) Return to original docking point (maintaining mechanical contact)
- 5) Closure of covers.

If both sides of the RM have to be serviced during the mission, the above phases will be repeated twice, of course between the two cycles a further exchange of docking points is necessary.
For sake of completion, it has been also evaluated the possibility of a non standard ORU exchange (in the SUPER-ORU area). In general the arms and rigidizer trajectories are not laid in a plane. Nevertheless, the considerations here

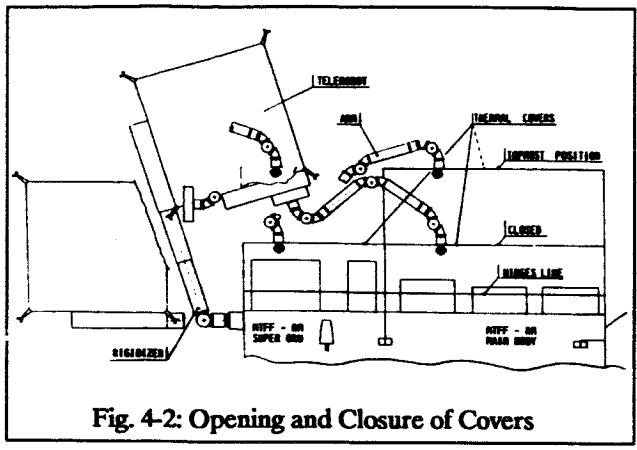


Fig. 4-2: Opening and Closure of Covers

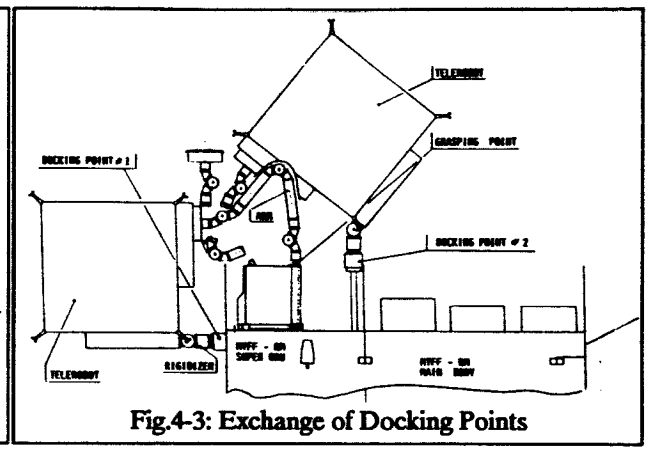


Fig.4-3: Exchange of Docking Points

developed are also applicable to skewed configuration; this fact can be stressed considering that the exact definition of the type and of the d.o.f number is well beyond the scope of this activity.

Phases 1 and 5: Opening and Closure of Covers.

These phases are summarized in **Figure 4-2.** and two different sequencies are foreseen:

- Opening and closure of SUPER-ORU area covers: in this case the rigidizer is only partially extracted.
- Opening and closure of standard ORU area (Main Body) covers: the rigidizer has its maximum length (2 m.) and its rotation allows one arm to reach the cover handles.

In the figure are shown the starting and the topmost positions (of the opening sequence).

Phases 2 and 4: Exchange of Docking Points.

These phases are summarized in **Figure 4-3.** One arm grasps a proper ORU grapple fixture, which becomes the pivot for the vehicle displacement. It has be noted that all the twist joints except for the one nearest to the grapple fixture, must have a rotation of 180° to complete the operation. At the end, the vehicle has a specular configuration with respect to the starting condition.
A rotation of the rigidizer twist joint around the axes of the new docking point can enable the system to service standard ORUs.

Phase 3: Standard ORUs Exchange.

The main features of this phase are summarized in **Figure 4-4** and it is possible to distinguish two different steps:

- To reach the ORUs and to extract them from RM
- To insert the ORUs in the body of the servicing vehicle in its own interfaces.

For the first step it is necessary to foresee two different arm positions (depending on ORUs distance). Anyway, there is just only one final position for the first step that is the starting position for the second one.
The second step is the most critical one, because of the arm dimensions: the arms length must be in the order of 2.5 (m).

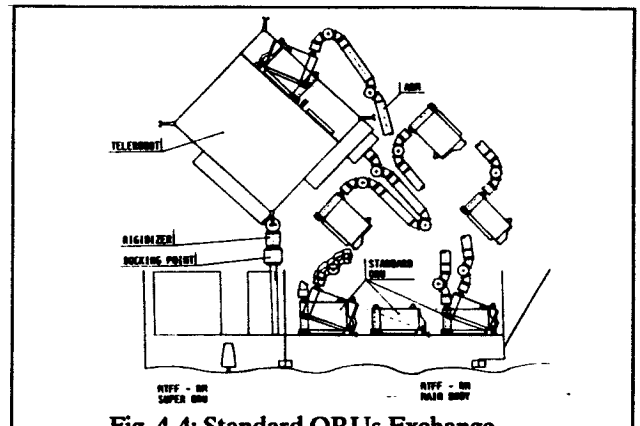


Fig. 4-4: Standard ORUs Exchange

Extraordinary Mission: Non Standard-ORU Exchange (in SUPER- ORU area).

This mission is summarized in **Figure 4-5** , except for the opening/clousure of covers. A final decision about non standard-ORUs shapes, dimensions and masses has not been yet estabilished.

For this reason this figure must be seen in a general sense: it has been considered one non standard ORU of 350 Kg. (5 times the maximum standard ORU mass), with a height which does not interfere with the cover, and with the other dimensions proportional to the standard ORU ones.
Within this data range it is possible to get a reasonable feeling of mission feasibility.

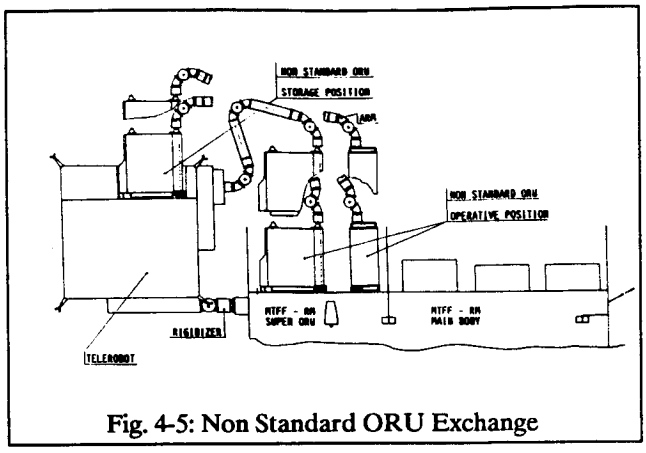


Fig. 4-5: Non Standard ORU Exchange

Manipulative S/S main features:

The manipulative S/S is schematized in **Figure 4-6**; it consists of the following main components:

- 2 arms (7 d.o.f. each)
- 1 rigidizer (3 d.o.f.)
- Drive Electronics Units
- (Robot Computer)
- (Robotic Sensors).

Figure 4-7 illustrates the main components features.

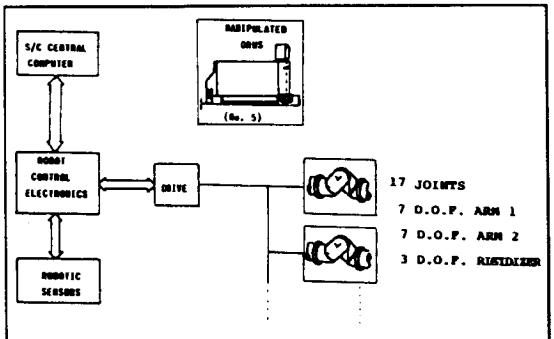


Fig. 4-6: Manipulative S/S Schematization

Arms

A number of possible arm configurations can be taken as valid. The one considered has been investigated in different activities and presents interesting features as far as workspace envelop is concerned. The joint torque level is in the 20-100 (Nm) range. The final articulation architecture, including joint selection, will be object of a dedicated technological activity.
Some technological difficulties are expected for the joint compactness requirements.

Rigidizer

The same considerations done for the arms apply to the rigidizer. It does not need a sophisticated control because its main scope is to position in relative way the Telerobot body and the ORUs layout. The joint torque level is 100 (Nm).

Drive Electronics.

It converts the command signals coming from the Robot Control Electronics into input currents to all the joint motors. It consists of two main sections: Power and Data I/F, and drive. No technological difficulties are foreseen except for problems related to dimension reduction (for example if the electronic is placed inside limbs).

Robotic Electronics and Robotic Sensors.

These items will be considered in the prosecution of the activity.

Manipulative S/S Overall Budget:

The manipulative S/S overall budget is summarized in **Table 4-1** with respect to mass, volume, energy requirements and expected technological risks.
The energy budget has been developed considering reference worst cases, with proper margins included during the process.
Assuming an EE/Spacecraft relative velocity in the order of 0.01 (m/s) and accounting the torque deliverable by

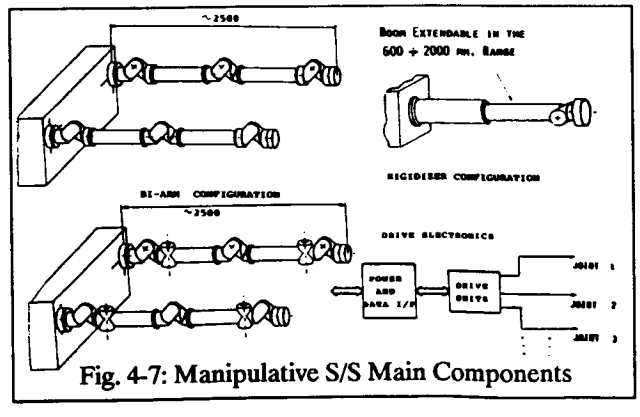


Fig. 4-7: Manipulative S/S Main Components

FEATURE / ITEM	MASS (Kg)	VOLUME (CM)	ENERGY (Joule)	SINGLE ITEM TECHNOLOGICAL RISKS
ARM 1	40	"SPECIAL LOCATION"	63.000	LIMITED TO JOINT COMPACTNESS
ARM 2	40	"SPECIAL LOCATION"		LIMITED TO JOINT COMPACTNESS
RIGIDIZER	30	"SPECIAL LOCATION"	32.000	LIMITED TO JOINT COMPACTNESS
MANIPULATED ORUS	350	1.6	TBD	N A
DRIVE	10	0.015	20.000	LIMITED TO COMPACTNESS (IN CASE OF SPECIAL LOCATION)
TOTAL (Including +5% margin)	≃ 492	≃ 1.7	≃ 120.000	

Tab. 4-1: Manipulative S/S Overall Budget

	MASS (Kg)	VOLUME (CM)	ENERGY (Joule)
COMPUTED PROPULSION PLUS MANIPULATIVE S/Ss FEATURES (INCLUDING FUEL AND 350 Kg OF ORUs)	634	2	375,000
ASSUMED REFERENCE TELEROBOT FEATURES (INCLUDING FUEL AND 350 Kg OF ORUs)	1,500	4.9	TBD
AVAILABILITY FOR REMAINING S/Ss (COMPUTING, SENSORS, BATTERIES, TTC, THERMAL, STRUCTURE, ETC....)	866	2.9	TBD

Tab. 5-1: Propulsion and Manipulative S/S Overall Features

joints as well as the full geometric sequence, the overall time required to fulfill the complete manipulative mission is in the order of 15,000 sec.
The 0.01 (m/s) relative velocity has been assumed based on the necessity to avoid any collision, case of failures, between whatsoever structure and the moving parts, with a velocity which can constitute a serious hazard.

5. CONCLUSIONS

A preliminary evaluation of the results obtained with respect to the propulsion and manipulative subsystems, indicates that Mass, Volume and Energy requirements, are in line with the general feature assumed for the Telerobot vehicle (see Table 5-1).
Moreover, from the technological risk point of view it seems that no special problems should arise, apart some electromechanical compactness requirements on typical space robotic components (joints, drivers).
Nevertheless, the confirmation on feasibility can be obtained only after completion of the remaining activities and,in this respect, particular importance will be given to the computing and TTC aspects.

REFERENCES

[1] M.H. Kaplan, "Modern Spacecraft Dynamics and Control", John Wiley & Sons, 1976

[2] F. Graziani, "Astrodinamica Applicata", Course Notes of the Post Graduated Aerospace School, Rome, 1984

[3] NASA, "SSP-30000 Sects. 3,4", Trajectory and Spacecraft design for Automated Approaches to the USSS.

[4] D. Long, "Characteristics of STS Earth Orbit Rendex-vous", NASA JSC, 1987

[5] Martin Marietta, "Manned Manoeuvring Unit", Technical and Operational Report, 1985

[6] F. Hechler, "Safe and Fuel Minimum reference Trajectories for Closed Loop Controlled Approaches", ESOC, 1987

[7] M.C. Eckstein, "Safe Rendez-vous Approach to a Space Station by Impulsive Transfer and Continuous Thrust Arcs", DFVLR, 1987

[8] P.G. Magnani, A. Gallo, "Sub-optimized Rendez-vous Approach Sequence", Post Graduated Aerospace School, Rome, 1987

[9] Tecnospazio S.p.A, "Telerobot Feasibility 1", Study 1988

[10] Tecnospazio S.p.A, "Project for Robotic, Teleoperated, and Upgradable Servicing (PROTEUS)", Study, 1987

[11] Italian Space Agency, "The Spider System", Definition Study, 1988

[12] Tecnospazio S.p.A, "Bracci Robotici e Sistemi di Manipolazione", Study, 1988

[13] Dornier, "RM Design Definition Report", Phase B2-Ext, Final Reviw report, 1988.

[14] British Aerospace, "Servicing Review", "EVA and Robotic Servicing Aids Study", 1988.

MODELING AND SENSORY FEEDBACK CONTROL FOR SPACE MANIPULATORS

Yasuhiro MASUTANI, Fumio MIYAZAKI, and Suguru ARIMOTO

Department of Mechanical Engineering
Faculty of Engineering Science, Osaka University
Toyonaka, Osaka, 560 JAPAN

Abstract

This paper treats the positioning control problem of the endtip of space manipulators whose base are uncontrolled. In such a case, the conventional control method for industrial robots based on a local feedback at each joint is not applicable, because a solution of the joint displacements that satisfies a given position and orientation of the endtip is not decided uniquely. We propose a sensory feedback control scheme for space manipulators based on an artificial potential defined in a task-oriented coordinates. Using this scheme, the controller can easily determine the input torque of each joint from the data of an external sensor such as a visual device. Since the external sensor is mounted on the unfixed base, the manipulator must track the moving image of the target. in the sensor coordinates. Moreover the dynamics of the base and the manipulator is interactive. However, we can prove that the endtip asymptotically approaches the target stationary in an inertial coordinate frame by the Liapunov's method. Finally results of computer simulation for a 6-link space manipulator model show the effectiveness of the proposed scheme.

1 Introduction

For the next generation of the space activities, various missions in satellite orbits are planned. In order to carry out these missions, robots that replace astronauts are indispensable, because the number of astronauts living in the orbit is limited. In particular, many plans of robots for extra vehicle activities have been proposed such as shown **Fig.1**. They are small spacecrafts(or satellites) equipped with manipulators and visual devices, and are thought to perform a wide variety of tasks while flying freely around the mother ships.

One of the differences of space manipulators from conventional ones on the ground is that their bases, namely spacecrafts, are unfixed. Therefore the degree-of-freedom of motion of the whole system increases by 6. When the endtip is positioned at a target object floating independently of the manipulator system, one may think that the base as well as the joints must be controlled. Of course thrusters and reaction wheels are necessary to translate and rotate bases themselves. Thrusters can output gas jet like on-off and reaction wheels can make moments by changing their angular velocities. In case that they cooperate with joint actuators to control the endtip of the manipulator, their capacities need to be large, or manipulators need to be operated slowly. Hence, it is inadequate to control the base and joints at the same time from the view point of energy, weight, or task efficiency. Therefore it is desirable to be able to control the endtip of manipulator only by the joint actuators, nevertheless the uncontrolled base is moved by the reactions from the manipulator.

When the base is uncontrolled, the base's motion depends on the action of joint. In such a case, the linear and angular momenta of the whole system floating in non-gravitational environment are conserved, because the joint actuators generate no external forces. Considering this constraint, the kinematic degree-of-freedom of the system reduces to the number of the joints. The position and orientation of the endtip, however, depend on the time history of joint displacements. In other words, they cannot be expressed as functions only of the joint displacements at each time. If one applied a conventional local feedback method to this system, one could not define the joint displacements corresponding to a given position and orientation of the endtip. Therefore the positioning control of the space manipulator needs global feedbacks using data of external sensors such as visual devices.

For this problem, Vafa and Dubowsky developed the *virtual manipulator concept*, that is an idealized kinematic chain [1]. When the base's attitude is controlled and its position is uncontrolled, this concept is very useful. Alexander and Cannon showed that the endtip can be controlled by solving inverse dynamics that includes the base's motion [2]. However, much calculation is necessary to get the control inputs by their method. Umetani and Yoshida proposed *the generalized Jacobian matrix*, that relates the endtip velocities to joint velocities by taking the base's motion into consideration [3]. They also showed that a desired endtip velocity can be resolved to joint velocities by this relation. However, they treated only the kinematic problem.

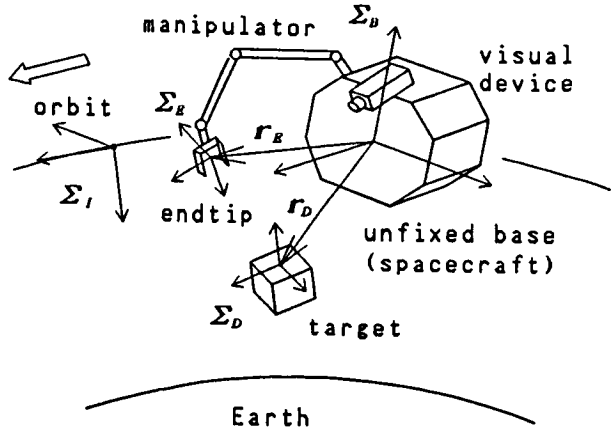


Fig.1: Space robot

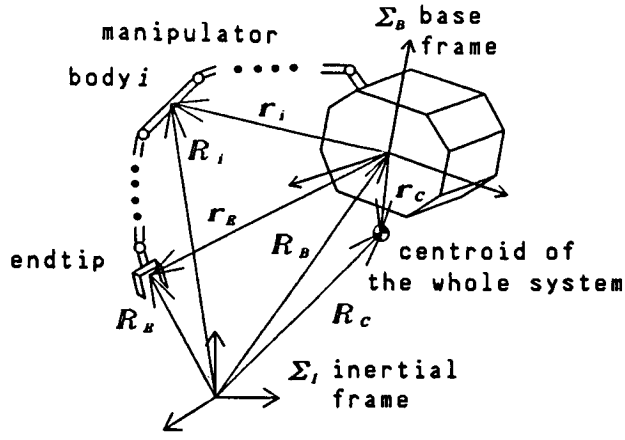


Fig.2: Model of the space manipulators

We propose a sensory feedback control scheme that position the endtip at the target specified in an inertial space without using inverse dynamics. This is based on the concept of an artificial potential defined in a task-oriented coordinate system [4,5]. Using this scheme, the deviations of the endtip measured with external sensor on the base is fed back to the each joint multiplied by the transpose of the Jacobian. Since the external sensor is mounted on the free-motion base, the sensor signal indicates as if the target were moving. Thus the manipulator must track the moving target, though this target is stationary in the inertial coordinate frame in fact. In general, it is difficult to show the asymptotic stability of the position control system tracking a moving point. Fortunately it is possible to prove it by using the Liapunov's method and the dynamical relationship of the motion between the base and joints.

In this paper, first we discuss the kinematics of space manipulators, and an algorithm to derive the generalized Jacobian from the conventional Jacobian is shown. Next, considering the dynamics of space manipulators, we represent the new control scheme, and discuss its stability. Finally we show the effectiveness of the proposed scheme through the computer simulation.

2 Kinematics

In this section, we consider kinematic relation between space manipulators and their bases by using the conservation laws of linear and angular momenta of the whole system.

2.1 Modeling

A space manipulator system in the earth's satellite orbit can be approximately considered to be floating in the non-gravitational environment. We treat this system as a set of $n+1$ rigid bodies connected with n joints that form a tree configuration. Each joint is numbered in series of 1 to n, and these displacements are represented by the joint vector $q = [\ q_1, q_2, \cdots, q_n\ ]^T$. On the other hand, each body is numbered in series of 0 to n. In particular, we assign B to the number of the base body. The mass and inertia tensor of i-th body are m_i and I_i respectively.

Let us define two coordinate frames. One is the inertial coordinate frame, Σ_I, that in the orbit where the system exists; another is the base coordinate frame, Σ_B, that is attached on the base, whose origin is located at the centroid of the base. In this paper, we express all vectors in terms of frame Σ_B. The reasons why we introduce the base coordinate frame are to contrast the properties of space manipulators with that ground-fixed manipulators and to use the data measured in this frame for the control scheme later. In order to represent the orientation of frame Σ_B relative to frame Σ_I, we use three appropriate parameters such as roll, pitch, and yaw and their vector form representation $\phi \in R^3$.

As shown in **Fig.2**, let $\boldsymbol{R}_i$ and $\boldsymbol{r}_i$ be position vectors pointing the centroid of the i-th body with reference to frames Σ_I and Σ_B respectively. The relation between them can be written by

$$\boldsymbol{R}_i = \boldsymbol{R}_B + \boldsymbol{r}_i. \tag{1}$$

where $\boldsymbol{R}_B$ is the position vector pointing the centroid of the base with reference to Σ_I.

Moreover, let $\boldsymbol{V}_i$ and $\boldsymbol{\Omega}_i$ be linear and angular velocities with reference to frame Σ_I, $\boldsymbol{v}_i$ and $\boldsymbol{\omega}_i$ be linear and angular velocities with reference to frame Σ_B. They have relations as follows;

$$\boldsymbol{V}_i = \boldsymbol{v}_i + \boldsymbol{V}_B + \boldsymbol{\Omega}_B \times \boldsymbol{r}_i, \tag{2}$$

$$\boldsymbol{\Omega}_i = \boldsymbol{\omega}_i + \boldsymbol{\Omega}_B, \tag{3}$$

where $\boldsymbol{V}_B$ and $\boldsymbol{\Omega}_B$ are linear and angular velocities of the centroid of the base with reference to frame Σ_I , and operator '$\times$' represents outer product of vectors. The Jacobian matrices corresponding to the linear and angular velocities of the centroid of each body can be obtained as functions of the joint vector $\boldsymbol{q}$;

$$\boldsymbol{v}_i = \boldsymbol{J}_{Li}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \tag{4}$$

$$\boldsymbol{\omega}_i = \boldsymbol{J}_{Ai}(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \tag{5}$$

2.2 Linear and Angular Momenta of the System

Since we treat in this paper the case in which no gravitational forces act and no actuators generating external forces are employed, the linear and angular momenta of the whole system are conserved. We assume here that the system is stationary in initial state. Thus these momenta are always zero. Using the relations given in the previous sub-section, we express these momenta as a linear combinations of $\boldsymbol{V}_B$, $\boldsymbol{\Omega}_B$, and $\dot{\boldsymbol{q}}$.

The linear momentum, $\boldsymbol{P}$, and angular momentum, $\boldsymbol{L}$, of the whole system are defined by;

$$\boldsymbol{P} \stackrel{\text{def}}{=} \sum_{i=0}^{n} m_i \boldsymbol{V}_i, \tag{6}$$

$$\boldsymbol{L} \stackrel{\text{def}}{=} \sum_{i=0}^{n} \{{}^B I_i \boldsymbol{\Omega}_i + m_i \boldsymbol{R}_i \times \boldsymbol{V}_i\}, \tag{7}$$

where ${}^B I_i$ means the inertia tensor in term of the base frame, Σ_B. Let us define the followings for the centroid of the whole system;

$$m_C \stackrel{\text{def}}{=} \sum_{i=0}^{n} m_i, \tag{8}$$

$$\boldsymbol{r}_C(\boldsymbol{q}) \stackrel{\text{def}}{=} \sum_{i=0,\ i\neq B}^{n} m_i \boldsymbol{r}_i(\boldsymbol{q})/m_C, \tag{9}$$

$$\boldsymbol{J}_C(\boldsymbol{q}) \stackrel{\text{def}}{=} \sum_{i=0,\ i\neq B}^{n} m_i \boldsymbol{J}_{Li}(\boldsymbol{q})/m_C \quad \in R^{3\times n}. \tag{10}$$

Substituting equations(1),(2), (3), (4),and (5) into equations(6) and (7) yields

$$\begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{L} \end{bmatrix} = \begin{pmatrix} \boldsymbol{H}_V & \boldsymbol{H}_{V\Omega} \\ \boldsymbol{H}_{V\Omega}{}^T & \boldsymbol{H}_\Omega \end{pmatrix} \begin{bmatrix} \boldsymbol{V}_B \\ \boldsymbol{\Omega}_B \end{bmatrix} + \begin{pmatrix} \boldsymbol{H}_{Vq} \\ \boldsymbol{H}_{\Omega q} \end{pmatrix} \dot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{R}_B \times \boldsymbol{P} \end{bmatrix}. \tag{11}$$

Each block is defined by

$$\boldsymbol{H}_V \stackrel{\text{def}}{=} m_C U_3 \quad \in R^{3\times 3}, \tag{12}$$

$$\boldsymbol{H}_{V\Omega} \stackrel{\text{def}}{=} -m_C[\boldsymbol{r}_C\times] \quad \in R^{3\times 3}, \tag{13}$$

$$\boldsymbol{H}_{Vq} \stackrel{\text{def}}{=} m_C \boldsymbol{J}_C \quad \in R^{3\times n}, \tag{14}$$

$$H_\Omega \stackrel{\text{def}}{=} \sum_{i=0,\ i\neq B}^{n} \{{}^B I_i + m_i D(\boldsymbol{r}_i)\} + I_B \quad \in R^{3\times 3}, \tag{15}$$

$$H_{\Omega q} \stackrel{\text{def}}{=} \sum_{i=0,\ i\neq B}^{n} \{{}^B I_i J_{Ai} + m_i [\boldsymbol{r}_i \times] J_{Li}\} \quad \in R^{3\times n}, \tag{16}$$

where U_3 is 3×3 unit matrix, and two matrix functions $[\boldsymbol{r}\times]$ and $D(\boldsymbol{r})$ for a vector argument $\boldsymbol{r} = [\ r_x,\ r_y,\ r_z\ ]^T$ are defined as follows;

$$[\boldsymbol{r}\times] \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \quad \in R^{3\times 3}, \tag{17}$$

$$D(\boldsymbol{r}) \stackrel{\text{def}}{=} [\boldsymbol{r}\times]^T\ [\boldsymbol{r}\times] \quad \in R^{3\times 3}. \tag{18}$$

Since the zero linear and angular momenta are assumed, substituting $\boldsymbol{P} \equiv \boldsymbol{O}$ and $\boldsymbol{L} \equiv \boldsymbol{O}$ into equation(11) and solving it for $\boldsymbol{V}_B$ and $\boldsymbol{\Omega}_B$, we obtain

$$\begin{bmatrix} \boldsymbol{V}_B \\ \boldsymbol{\Omega}_B \end{bmatrix} = \begin{pmatrix} -J_C(\boldsymbol{q}) - [\boldsymbol{r}_C(\boldsymbol{q})\times] H_B{}^{-1}(\boldsymbol{q}) H_M(\boldsymbol{q}) \\ -H_B{}^{-1}(\boldsymbol{q}) H_M(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}}, \tag{19}$$

where

$$H_B \stackrel{\text{def}}{=} H_\Omega - m_C D(\boldsymbol{r}_C) \quad \in R^{3\times 3}, \tag{20}$$

$$H_M \stackrel{\text{def}}{=} H_{\Omega q} - m_C [\boldsymbol{r}_C \times] J_C \quad \in R^{3\times n}. \tag{21}$$

Matrix H_Ω and H_B are the inertia tensors of the whole system with respect to rotation about the centroid of *the base*, $\boldsymbol{R}_B$, and the centroid of *the whole system*, $\boldsymbol{R}_C$. The inverse of inertia tensor H_B always exists, because it is positive definite. Equation(19) is the significant and basic relation for space manipulators that represents how the base is translated and rotated($\boldsymbol{V}_B$, $\boldsymbol{\Omega}_B$) by the joint action($\dot{\boldsymbol{q}}$).

Between angular velocity of the base, $\boldsymbol{\Omega}_B$, and time derivative of the orientation parameters of the base, $\dot{\boldsymbol{\phi}}$, there is a relation

$$\boldsymbol{\Omega}_B = N(\boldsymbol{\phi})\ \dot{\boldsymbol{\phi}}, \qquad N(\boldsymbol{\phi})\ \in R^{3\times 3}. \tag{22}$$

From equation(19)and(22), we can derive

$$N(\boldsymbol{\phi})\ \dot{\boldsymbol{\phi}} = -H_B{}^{-1}(\boldsymbol{q}) H_M(\boldsymbol{q})\ \dot{\boldsymbol{q}}. \tag{23}$$

Since this equation is not integrated, the orientation of the base depends on the time history of the joint variables. Therefore the position and orientation of the endtip cannot be expressed as functions only of the joint variables, $\boldsymbol{q}$. Conversely even though the position and orientation of the endtip are given, it is impossible to obtain the joint variables to satisfy the given conditions in the way industrial robots always do. This is the reason why the global control feeding back information of the endtip is required.

2.3 Generalized Jacobian Matrix

Applying equation (19), the linear and angular velocities, $\boldsymbol{V}_E$ and $\boldsymbol{\Omega}_E$, can be expressed as a linear combination only of the joint velocities. The equations with respect to the endtip motion such as equations(2), (3), (4), and (5) are hold, thus substituting equations(19) into them yield

$$\boldsymbol{V}_E = \hat{J}_L(\boldsymbol{q})\ \dot{\boldsymbol{q}}, \tag{24}$$

$$\boldsymbol{\Omega}_E = \hat{J}_A(\boldsymbol{q})\ \dot{\boldsymbol{q}}, \tag{25}$$

where

$$\hat{J}_L \stackrel{\text{def}}{=} J_L - J_C + [(\boldsymbol{r}_E - \boldsymbol{r}_C)\times]\ H_B{}^{-1} H_M \quad \in R^{3\times n}, \tag{26}$$

$$\hat{J}_A \stackrel{\text{def}}{=} J_A - H_B{}^{-1} H_M \quad \in R^{3\times n}. \tag{27}$$

In equations(26), vector r_E is the position of the endtip with respect to the base. Matrices J_L and J_A are respectively *the conventional Jacobian matrices* that correspond to the linear and angular velocities relative to the base. On the other hand, matrices $\hat{J}_L$ and $\hat{J}_A$ are called *the generalized Jacobian matrices.* They are associated with the linear and angular velocities including the influence of the base motion.

Both V_E and Ω_E what we use here are relative to *the inertial coordinate frame*, Σ_I, and they are expressed in terms of *the base coordinate frame*, Σ_B. This representation of velocities is unnatural, because the value of them cannot be directly derived from the data of the endtip measured with the sensor on the base. Using them, however, the generalized Jacobians, $\hat{J}_L$ and $\hat{J}_A$, can be expressed as functions only of joint variable q, and this clarifies the differences from the conventional Jacobian. Equations(26)and(27)show that the generalized Jacobian is summation of the conventional one and additional terms, and the additional term depend on the mass and inertia tensor of the each body as well as their dimensions. If the inertia of the base is very large, namely $m_B \to \infty, I_B \to \infty$, the additional term becomes zero and the generalized Jacobian is equal to the conventional one, because $J_C \to O$, $r_C \to O$, and $H_B \to \infty$.

3 Dynamics

In this section, we derive the equation of motion that is required to discuss the stability of the control schemes proposed later by using the Lagrangian formulation.

3.1 Kinetic Energy

The total kinetic energy of the whole system is defined by summing up the translational energy and rotational energy of each body.

$$T \stackrel{\text{def}}{=} \frac{1}{2}\sum_{i=0}^{n}\{\Omega_i{}^T\, {}^B I_i \Omega_i + m_i V_i{}^T V_i\}. \tag{28}$$

Substituting equation(28) into equations(2)and(3) yields

$$T = \frac{1}{2}[\, V_B{}^T \;\Omega_B{}^T \;\dot{q}^T\,] \begin{pmatrix} H_V & H_{V\Omega} & H_{Vq} \\ H_{V\Omega}{}^T & H_\Omega & H_{\Omega q} \\ H_{Vq}{}^T & H_{\Omega q}{}^T & H_q \end{pmatrix} \begin{bmatrix} V_B \\ \Omega_B \\ \dot{q} \end{bmatrix}, \tag{29}$$

where matrix H_q is given by

$$H_q \stackrel{\text{def}}{=} \sum_{i=0,\ i\neq B}^{n} \{J_{Ai}{}^T\, {}^B I_i J_{Ai} + m_i J_{Li}{}^T J_{Li}\} \quad \in R^{n\times n} \tag{30}$$

that is the inertia matrix of the manipulator whose base is fixed [6].

In case that the linear and angular momenta are conserved at zero, the relevant equation(19) holds, and substituting it into equation(29), we obtain the reduced form

$$T = \frac{1}{2}\dot{q}^T \hat{H}(q)\, \dot{q}, \tag{31}$$

where

$$\hat{H} \stackrel{\text{def}}{=} H_q - m_C J_C{}^T J_C - H_M{}^T H_B^{-1} H_M \quad \in R^{n\times n}. \tag{32}$$

Using the fact that the $(n+6)\times(n+6)$ original inertia matrix in equation (29) is positive definite, it is easy to prove the reduced inertia matrix, $\hat{H}$, to be symmetric and positive definite. Moreover each entry of $\hat{H}$ is a function only of the joint variable, q. Therefore the equation of motion for the reduced form can be derived in the same way used in conventional robotics.

3.2 Equation of Motion

Since there is no potential energy in non-gravitational environments, the Lagrangian is equal to the kinetic energy. Using the Lagrangian formulation, we can derive the equation of motion of the system

$$\hat{H}\ddot{q} + \dot{\hat{H}}\dot{q} - \frac{\partial}{\partial q}\{\frac{1}{2}\dot{q}^T\hat{H}\,\dot{q}\} = \tau, \tag{33}$$

where vector $\tau \stackrel{\text{def}}{=} [\ \tau_1, \tau_2, \cdots, \tau_n\ ]^T$ is composed of the control inputs into joint actuators.

4 Sensory Feedback Control

In this section, a new positioning control scheme for the endtip of manipulator whose base is uncontrolled during manipulation.

The problem here is to position the endtip of manipulator at a target on an object floating still in the inertial frame. It is assumed that the position and attitude of the base is initially arranged so that the endtip of the manipulator can reach the target.

4.1 Deviation of the Endtip

While the base is moved by the reaction force and moment from the manipulator, the position and orientation of the target seen from the view point on the base change. In order to represent the orientation of a certain coordinate frame Σ_*, we introduce a matrix A_* that plays the role of rotational coordinate transformation from frame Σ_* to Σ_B. This matrix consists of the three unit vectors along the coordinate axes of Σ_*, namely n_*, s_*, and a_* orthogonal to each other;

$$A_* \stackrel{\text{def}}{=} [\ n_*\ s_*\ a_*\ ] \quad \in R^{3\times 3}. \tag{34}$$

Let r_D and A_D be respectively the position and orientation of the target that the endtip is desired to approach, r_E and A_E be respectively the position and orientation of the endtip. One can easily derive them from the data of an external sensor such as a vision device mounted on the base.

Now let us define position and orientation deviations, e_p and e_o, which represent the difference in position and orientation between the target and the endtip of the manipulator;

$$e_p \stackrel{\text{def}}{=} r_D - r_E \quad \in R^3 \tag{35}$$

$$e_o \stackrel{\text{def}}{=} \frac{1}{2}(n_E \times n_D + s_E \times s_D + a_E \times a_D) \quad \in R^3. \tag{36}$$

They are employed in the control scheme of the next sub-section. The orientational deviation, e_o, is the same as Luh's introduced in resolved-acceleration control [7].

4.2 Control Scheme

We represent a control scheme that feeds back the above-mentioned deviations to the input torque of each joint τ, which is given by

$$\tau = k_p\ \hat{J}_L^T(q)\ e_p + k_o\ \hat{J}_A^T(q)\ e_o - K_V\ \dot{q}, \tag{37}$$

where k_p and k_o are the positive scalar gains for the position and orientation respectively, and symmetric positive definite $K_V \in R^{n\times n}$ is the gain matrix for joint velocities. The block diagram of this scheme is shown in **Fig.3**. In this scheme, the controller can determine easily the inputs torque of each joint from the deviations of the endtip, e_p and e_o, multiplied by gains k_p and k_o and the transpose of the generalized Jacobian $\hat{J}_L$ and $\hat{J}_A$. Moreover the position and attitude of the base is not required to measure, because they are never used in the whole process of deriving the control input.

Since the external sensor is mounted on the free-motion base, the manipulator must track the moving target that is actually floating stationary in inertial space. Generally it is difficult to discuss the global stability of the tracking system of a moving target. In our case, however, we can prove the stability because the seeming

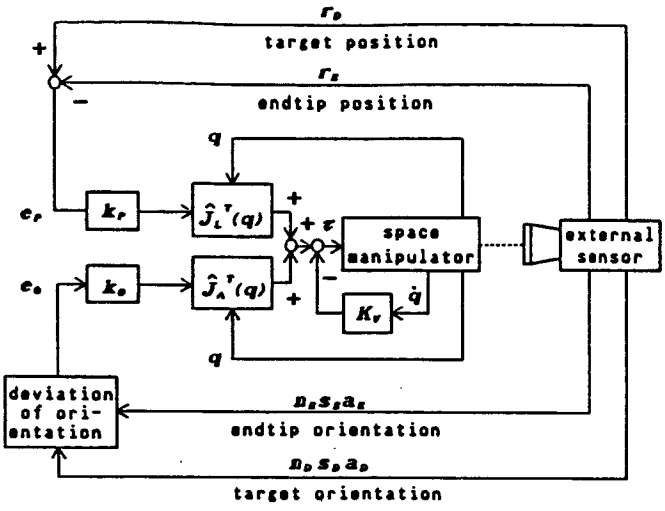


Fig.3: Block diagram of the control scheme

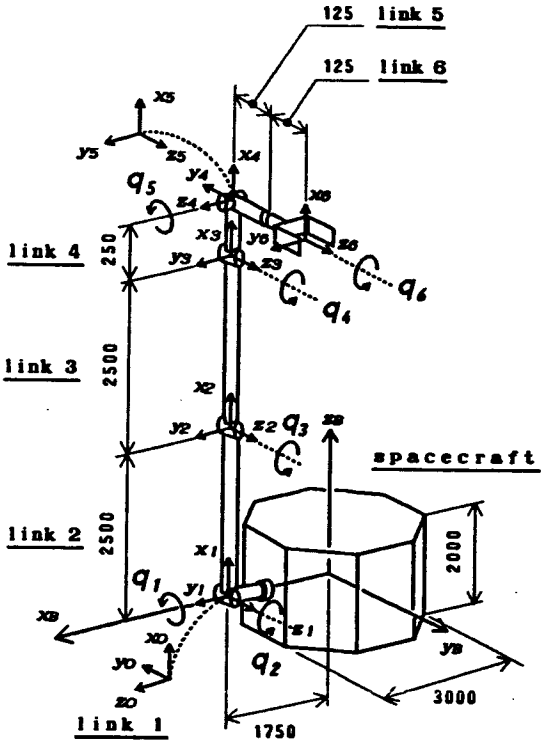


Fig.4: 6-link manipulator mounted on spacecraft

link		base	1	2	3	4	5	6
m_i	(kg)	2000	20.00	50.00	50.00	10.000	5.000	5.000
	x	1400	0.15	0.25	0.25	0.050	0.025	0.025
I_i	y	1400	0.10	26.00	26.00	0.075	0.025	0.025
(kgm^2)	z	2040	0.15	26.00	26.00	0.075	0.019	0.025

motion of the target object depends on only the motion of joints. If the following conditions hold during the manipulation,

$$\text{rank}\ (\ \hat{J}_L^T\ \hat{J}_A^T\ ) \geq 6, \tag{38}$$

$$\boldsymbol{n}_E{}^T\ \boldsymbol{n}_D + \boldsymbol{s}_E{}^T\ \boldsymbol{s}_D + \boldsymbol{a}_E{}^T\ \boldsymbol{a}_D > -1, \tag{39}$$

the closed loop system applied this scheme is asymptotically stable. In other words, the position and orientation of the endtip converge to the target as $t \to \infty$. It should be noted that the condition(39) means that the orientation of Σ_D is not diametrically directed to that of Σ_E.

Proof : To represent the whole system in our case, the state variable must include of the orientation the base $\boldsymbol{\phi}$ as well as $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ of the joint. Then the state variable vector $\boldsymbol{z}$ is defined as;

$$\boldsymbol{z} \stackrel{\text{def}}{=} [\ \boldsymbol{\phi}^T\ \boldsymbol{q}^T\ \dot{\boldsymbol{q}}^T\ ]^T \quad \in R^{2n+3}. \tag{40}$$

The closed loop system including the control scheme is written by the differential equation

$$\dot{\boldsymbol{z}} = \boldsymbol{F}(\boldsymbol{z}), \tag{41}$$

where function $\boldsymbol{F}(\boldsymbol{z})$ is defined by

$$\boldsymbol{F}(\boldsymbol{z}) \stackrel{\text{def}}{=} \begin{bmatrix} G(\boldsymbol{\phi}, \boldsymbol{q})\ \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \\ \hat{H}^{-1}(\boldsymbol{q})\ \boldsymbol{b}(\boldsymbol{\phi}, \boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \in R^{2n+3}, \tag{42}$$

$$G \stackrel{\text{def}}{=} -N^{-1}\ H_B^{-1}\ H_M \quad \in R^{3\times 3}, \tag{43}$$

$$b \stackrel{\text{def}}{=} -\dot{\hat{H}}\,\dot{q} + \frac{\partial}{\partial q}\{\frac{1}{2}\dot{q}^T \hat{H}\,\dot{q}\} + \tau \quad \in R^n. \tag{44}$$

For this proof, let us introduce another orientational deviation vector;

$$E_o \stackrel{\text{def}}{=} \begin{bmatrix} n_D - n_E \\ s_D - s_E \\ a_D - a_E \end{bmatrix} \quad \in R^9. \tag{45}$$

Now we propose a function of z

$$W(z) = \frac{1}{2}k_p\, e_p{}^T\, e_p + \frac{1}{4}k_o\, E_o{}^T\, E_o + \frac{1}{2}\dot{q}^T \hat{H}\,\dot{q} \tag{46}$$

as a candidate of Liapunov function. This function is zero in the set of desired state

$$\mathcal{E} \stackrel{\text{def}}{=} \{\ z \mid r_E = r_D,\ \ n_E = n_D,\ \ s_E = s_D,\ \ a_E = a_D,\ \ \dot{q} = O_n\}, \tag{47}$$

and is certainly positive except for the set $\mathcal{E}$. In space z, set $\mathcal{E}$ consists of not only an isolated point but also a set of connected points.

The time derivative of function W along the solution trajectory of equation(41) is

$$\dot{W} = -\dot{q}^T K_V \dot{q} \quad \leq 0, \tag{48}$$

that is, negative semi-definite in term of state z. To evaluate this equation, the following relations and the equation of the generalized Jacobian(24)and(25) are used;

$$\dot{q}^T \frac{\partial}{\partial q}\{\frac{1}{2}\dot{q}^T \hat{H}\,\dot{q}\} = \frac{1}{2}\dot{q}^T \dot{\hat{H}}\,\dot{q}, \tag{49}$$

$$\dot{e}_p = V_E - [\Omega_B \times]\, e_p, \tag{50}$$

$$\dot{E}_o = \begin{pmatrix} [n_E \times] \\ [s_E \times] \\ [a_E \times] \end{pmatrix} \Omega_E - \begin{pmatrix} [\Omega_B \times] & O & O \\ O & [\Omega_B \times] & O \\ O & O & [\Omega_B \times] \end{pmatrix} E_o, \tag{51}$$

If the time derivative of function $\dot{W}$ is equal to zero, $\dot{q} = O_n$, all states in the invariant set satisfy the following equation;

$$(\ \hat{J}_L^T\ \hat{J}_A^T\ ) \begin{bmatrix} k_p\, e_p \\ k_o\, e_o \end{bmatrix} = O_n, \tag{52}$$

$$\dot{\phi} = O_3. \tag{53}$$

Condition(38) suggests that equation(52) is equivalent to $e_p = O_3$ and $e_o = O_3$. Furthermore if condition(39) is satisfied, equation $e_o = O_3$ is equivalent to $E_o = O_9$. Therefore the maximal invariant set whose entry satisfies $\dot{W} = 0$ is equal to set $\mathcal{E}$. According to the theorem of LaSalle, set $\mathcal{E}$ of system(41) is asymptotically stable. □

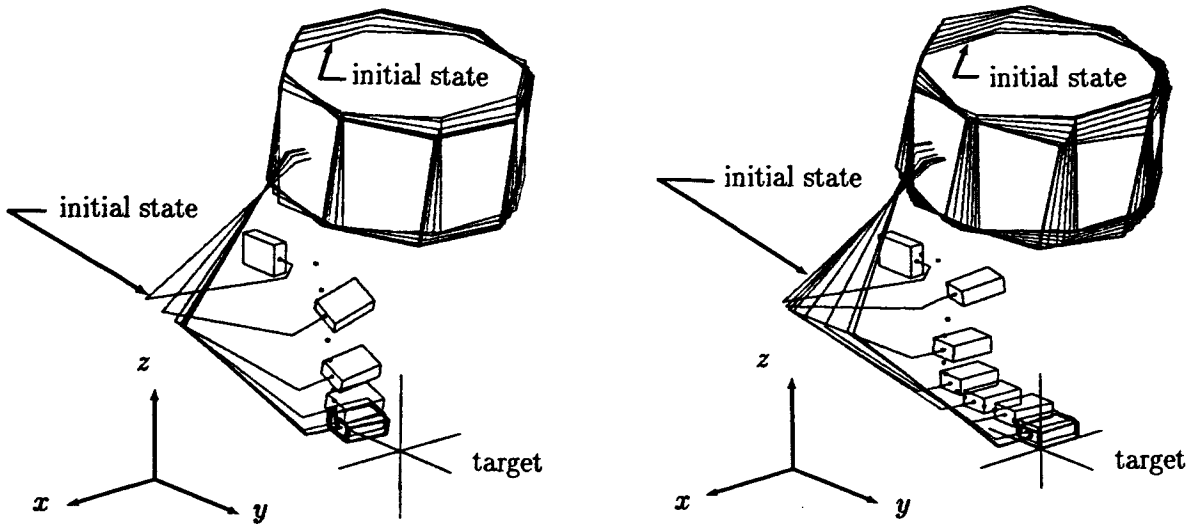
If conditions(38)and(39) are not satisfied, there are some equilibrium points except for the desired state in set $\mathcal{E}$, namely the singular points of Jacobian where the system will get stuck. However, this problem is not serious because the controller can make a temporary input to get the system out of the singular configuration.

Since this scheme is not a trajectory control, the endtip cannot always approach along a line to the target. However, making a virtual target on the line from the current endtip to the real target, the endtip moves nearly on the straight line. To realize this idea, one can use the following deviations $\bar{e}_p$ and $\bar{e}_o$ instead of e_p and e_o.

$$\bar{e}_p \stackrel{\text{def}}{=} \begin{cases} e_p & \|e_p\| \leq e_{pmax} \\ \dfrac{e_p}{\|e_p\|} e_{pmax} & \|e_p\| > e_{pmax} \end{cases} \tag{54}$$

$$\bar{e}_o \stackrel{\text{def}}{=} \begin{cases} e_o & \|e_o\| \leq e_{omax} \\ \dfrac{e_o}{\|e_o\|} e_{omax} & \|e_o\| > e_{omax} \end{cases} \tag{55}$$

Positive numbers e_{pmax} and e_{omax} are the maximal values of the norms of the deviations. This method does not influence the stability of the system.



(a) Local feedback scheme (b) Proposed scheme

Fig.5: Results of the simulations (state every 3 seconds)

5 Computer Simulation

We verify the effectiveness of the control scheme proposed in the previous section through the computer simulation. We use a 140(kg) weight manipulator model mounted on a 2000(kg) weight spacecraft shown in **Fig.4**, that has six revolute joints.

First a result that suggest the local feedback schemes are not suitable is shown in **Fig.5(a)**. In this figure, the endtip has a box to make its orientation clear. The endtip is positioned at a different point from the target, because the desired joint values are calculated on the assumption that the base is fixed. The inertia of the base in this model is quite large, however, the base's motion is not negligible for the endtip control.

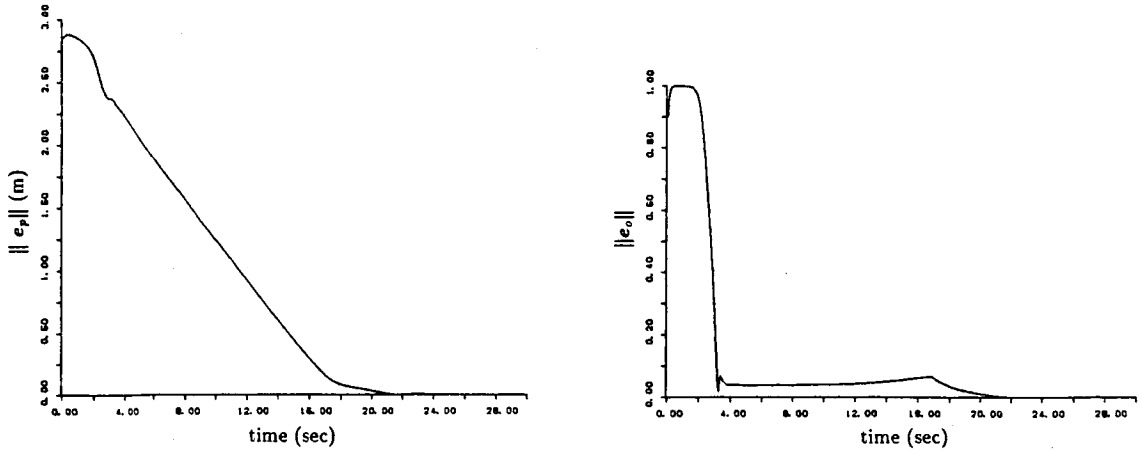
Next we show a result using the proposed scheme in case that the gains and the limits of deviations are given as;

$$\begin{aligned} k_p &= 75 \text{ (N)}, \\ k_o &= 100 \text{ (Nm)}, \\ K_V &= \text{diag } [400, 100, 200, 10, 2, 2] \text{ (Nmsec/rad)}, \\ e_{pmax} &= 0.2 \text{ (m)}, \\ e_{omax} &= 0.1. \end{aligned}$$

The state of the system is displayed every 3 seconds in **Fig.5(b)**. **Fig.6** represents the time responses of deviations $\|e_p\|$ and $\|e_o\|$. When the endtip reach the target, the translational displacement of the base is x: 0.01 (m), y: 0.07 (m), z: 0.04 (m) and rotational displacement of the base is roll: 16 (deg), pitch: 10 (deg), yaw: 1 (deg). The translational displacement is small, on the other hand, rotational one is so large that have a quite influence on positioning of the endtip. Using the proposed scheme, the position and orientation of the endtip reach the target, nevertheless the base is moved as this results.

6 Conclusion

For space manipulators, global control scheme that feed directly information of external sensors back to joints actuators is indispensable. In this paper, we proposed one method to realize them.



(a) Position (b) Orientation

Fig.6: Time responses of the norm of the endtip deviations

Computer technologies in space are developing slowly in comparison with those on the ground due to the influence of radiant rays and badness of heat radiation. Therefore the simple control scheme as equation(37) that does not burden computers is suited to use in space.

In this paper, we also show the method that makes up the generalized Jacobian from the conventional Jacobian. As shown in this process, obtaining the generalized Jacobian needs much more calculation than the conventional Jacobian. Moreover it needs identification of inertial parameters as well as dimensional parameters. To avoid these obstacles, we are examining the possibility to apply approximately the conventional Jacobians, J_L and J_A, in equation(37) instead of the generalized Jacobian, $\widehat{J}_L$ and $\widehat{J}_A$ [8].

References

[1] Z. Vafa and Z. Dubowsky. On the dynamics of manipulators in space using the virtual manipulator approach. In *Proceeding of the 1987 IEEE international Conference on Robotics and Automation*, pages 579–585, 1987.

[2] H.L. Alexander and R.H. Cannon. Experimental on the control of a satellite manipulator. In *Proceeding of American Control Conference*, 1987.

[3] Y. Umetani and K. Yoshida. Continuous path control of space manipulator mounted on OMV. *Acta Astronautica*, 15(12):981–986, 1987.

[4] M. Takegaki and S. Arimoto. A new feedback method for dynamic control of manipulators. *ASME Journal of Dynamics, System, Measurement, and Control*, 103:119–125, 1981.

[5] F. Miyazaki and S. Arimoto. Sensory feedback for robot manipulators. *Journal of Robotic Systems*, 2(1):53–71, 1985.

[6] H. Asada and J.J.E. Slotine. *Robot Analysis and Control.* Wiley-Interscience, 1985.

[7] J.Y.S. Luh, M.W. Walker, and P.R.C. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transaction on Automatic Control*, 25(3):468–474, 1980.

[8] F. Miyazaki, Y. Masutani, and S. Arimoto. Sensor feedback using approximate Jacobian. In *Proceedings of the USA-Japan Symposium on Flexible Automation*, pages 139–145, 1988.

CONTROL STRATEGIES FOR A TELEROBOT

John O'Hara[1] and Bill Stasi
Advanced Systems
Grumman Space Systems
Bethpage, New York 11714

Abstract

One of the major issues impacting the utility of telerobotic systems for space is the development of effective control strategies. For near-term applications, telerobot control is likely to utilize teleoperation methodologies with integrated supervisory control capabilities to assist the operator. In this paper, two experiments are reported which evaluated two different approaches to telerobotic control: bilateral force reflecting master controllers and proportional rate six degrees of freedom (DOF) hand controllers. The first experiment compared the controllers on performance of single manipulator arm tasks. The second experiment required simultaneous operation of both manipulator arms and complex multiaxis slave arm movements. Task times were significantly longer and fewer errors were committed with the hand controllers. The hand controllers were also rated significantly higher in cognitive and manual control workload on the two-arm task. The master controllers were rated significantly higher in physical workload. The implications of these findings for space teleoperations and higher levels of control are discussed.

1. INTRODUCTION

Automation and robotics (A&R) will play an important role in future space activities such as satellite servicing and space station assembly and maintenance. Two characteristics of space operations make it well suited to an A&R technology. First, space is an extremely hostile environment. To safely work in space, the extravehicular activity (EVA) astronaut must don an extravehicular mobility unit (EMU), which greatly reduces his dexterity and ability to work. Second, human resources in space are extremely limited, and the dollar cost is high. In addition, missions that are dangerous, require immediate action, or can be performed at orbits not convenient to shuttle operations will require a robotics capability for safe and efficient operations.

Space offers a challenge to robotics technology. Few operations in space are routine. A robotic system must, therefore, be capable of handling tasks that are either unplanned or cannot be planned to an assembly-line level of detail, i.e., the system must be capable of real-time strategy modification.

---

[1]Current address: Brookhaven National Laboratory, Building 130, Upton, New York, 11973.

Given present technology limitations, these requirements are somewhat beyond the capability of fully automated systems. It is desirable, therefore, to include the "man-in-the-loop" to integrate his knowledge base and decision-making capabilities into system operations. Thus, near term space robotic systems are likely to be teleoperated as an intermediate step to supervisory control and full automation.

Rice et al. [1] identified three factors in space teleoperations limiting human integration with the system. Included were the provision of adequate visual feedback from the worksite, the development and selection of control devices, and the effect of communication time delays on performance. This study is directed toward the issue of control input devices. A wide variety of control devices are presently used for teleoperating ranging from point-by-point position switches to hand controllers to master replica arms. The selection of the system is often governed by the unique uses of the manipulator. In space applications, however, manipulators are intended to be used for a vast array of different functions. Hence the selection of control input devices is especially difficult.

Recent developments in hand controller technology have enabled the integration of all six DOF into a single compact controller. Concern has been voiced over the ability of the operator to precisely control six DOF with one hand [2]. Inadvertent cross-coupling of axes is expected to be much more likely with a six-DOF controller. Heartly et al. [3] reported a comparison of a six-DOF controller with two three-DOF controllers on several manipulator tasks. The six-DOF controller was found to be useful for tasks requiring single axis commands and where workload was very low. For high-workload tasks where multiaxis inputs were required, the three-DOF controllers were found to be superior. Six-DOF controllers have been found to more mentally demanding than three-DOF controllers [1]. However, in a study where three- and six-DOF controllers were directly compared for operating a simulated Shuttle Orbiter remote manipulator system, no differences were found between the controllers in performance errors for low workload docking tasks or high workload tracking tasks [4]. The purpose of the present study is to extend the research on six-DOF controllers to dexterous manipulators.

Two studies were conducted evaluating methods of controlling a dexterous telerobot. In the first study, tasks requiring only single-arm control (six-DOF tasks) were performed including a task board and a simulated satellite orbital replacement unit (ORU) removal and installation. In the second study a truss assembly task requiring simultaneous control of both arms (a 12-DOF task) was performed. The main objective was to compare two types of controllers: 1) bilateral-force-reflecting master-arm replica controllers based upon joint position control; and 2) six-DOF ball-shaped hand controllers based upon proportional resolved-rate control. Since forces were not reflected to the hand controllers, an audio signal was provided to the operator which changed in tone and pulse frequency as forces built-up in the slave arms.

2. STUDY 1

2.1 Methods

2.1.1 Participants - Six subjects having little experience operating dexterous manipulators participated.

2.1.2 Experimental Design - Two factors were investigated: (1) control system having two levels (master controller and six-DOF hand controller); (2) task scenario having two levels (task board and ORU changeout). The variables were orthogonally combined and varied within subject. For certain analyses, the change in performance over trials was examined too.

2.1.3 Laboratory and Test Equipment - The study was conducted in Grumman's Telerobotics Development Laboratory. The facility was divided into a remote worksite, the telerobot systems and test articles in one area while the operator workstation, including all controls and displays, were located in another area. Direct visual contact with the worksite was prevented by a curtain.

The telerobot system consisted of a Teleoperator Systems Corporation SM-229 master-slave manipulator with Oak Ridge National Laboratory electronics and is owned by the Princeton Plasma Physics Laboratory. It was a bilateral-force-reflecting system with two six-DOF slave arms, plus one-DOF end effector. Each arm was capable of shoulder roll ($\pm$ 45°) and pitch ($\pm$ 90°), elbow pitch (+ 160°/50°) and yaw ($\pm$ 180°), and wrist roll (360°) and pitch (+ 45°/-120°).

Four CCTV cameras were used to relay visual information from the worksite to the operator's workstation. Two cameras were located to provide views from approximately 45° right and left of the manipulator. This camera was positioned approximately 0.76 m above the manipulator arms. The fourth camera was also located between the manipulator slave arms, but was positioned approximately 1.8 m above the arms. While the first three cameras could be controlled with pan, tilt, iris, and zoom functions, pan and tilt were not available for the fourth camera.

The workstation was designed from concepts developed for a generic workstation for the Space Station. The workstation was 211 cm high, 171 cm wide, divided into center, left, and right modules. The left and right modules were rotated 30° toward the center module. The center module housed a 48 cm low resolution CCTV monitor and the right module housed two 20 cm monitors. All camera controls were located on a panel to the left of the workstation. Since only three views of the worksite could be displayed at one time, the camera console provided a camera select function for choosing which camera views were presented.

The master controller was a full-scale replica of the slave arm. Control was achieved through position control loops for each DOF. The operator positioned the master arm to the desired configuration creating a position error between the master and slave. Motors in the salve arm drove it until the position error was zero. When zero position error could not be attained, force was reflected to the master arm. The grippers were closed and opened by squeezing the hand grip.

The six-DOF hand controllers were developed by CAE Electronics, Ltd. They provided bidirectional resolved rate control of the X, Y, and Z axes and roll, pitch, and yaw. Voltage outputs from the controllers were proportional to the displacement of the hand gripper ball and were transformed by computer to an end effector displacement vector (direction and velocity). All slave arm joint angles and velocities required to implement a control input were resolved in software. The spherical gripper ball was approximately 7.35 cm in diameter.

A fin was vertically positioned along the forward side of the ball to facilitate hand grip. The slave ar gripper was opened and closed by depressing the switches located on the forward side of the ball. The translation motion envelope provided for ± 95 cm of displacement from the centerpoint along each axis, and the rotational motion envelope provided for ± 15° for roll and pitch and ± 10° for yaw. When released, the controller returned to the center location. The hand controller processing computer provided rudimentary auditory force feedback information which was proportional to the amount of force in the slave. The hand controllers were operated in a "base frame" mode. The translational axes were referenced to a universal coordinate system, while the rotations were relative to the slave arm wrist orientation.

The test articles consisted of a task board and a simulated ORU. The task board provided the operator with a group of tasks requiring a wide variety of generic manipulator activities including grasping, translation, object rotation, hand-eye coordination, and dexterity. A peg was grasped with the end effector and used to perform subsequent tasks, such as activating a series of switches and plates. Indicator lights provided feedback to the operator upon successful task execution.

The surface-mounted ORU apparatus was a 51 x 51 x 76 cm box mounted on a wooden surface by a latch on its right and left sides, which was operated by rotating a metal handle. A center handle facilitated movement of the ORU which was counterweighed with a six-DOF off-loading system to simulate zero gravity conditions.

2.1.4 Procedures - Each subject was briefed on the use of the manipulator and the specific task scenarios. Each subject was also given time to operate the manipulator and several familiarization trials for each scenario prior to any test trail. The task board scenario required grabbing the peg from a tool rack with the slave arm, executing a fixed series of operations, and returning the peg to the tool rack. The ORU task required the unlatching, removal, storage, replacement, and securing of the ORU. All tasks were performed with a single slave arm. CCTV was used, and control of camera views and functions was achieved through subject's voice command to a laboratory assistant.

The task board scenario was accomplished first. Within each type of scenario, the sequence of controller types was balanced so that half the subjects used the master controller first. Four test trials of task board and two of the ORU scenario were performed.

2.1.5 Dependent Variables - Total task length was recorded for the task board, while both total time and subtask times were recorded for the ORU task. Two measures of manipulator control quality were obtained: error frequency and test conductor's evaluation of control/efficiency. Errors were defined as inappropriate contacts between the manipulator and the test article, termination and restart of a task sequence, and dropping of a tool or test article. During the ORU task, evaluations of the test subject's control of the manipulator were made to gauge the smoothness and efficiency of control actions (e.g., to distinguish between rough jerky movements and smooth, efficient movements). A five-point scale was established on which higher scores were indications of greater control.

Operator workload was evaluated along cognitive, manual control, and physical dimensions. An overall workload assessment was obtained as well. The workload evaluations were made by test subjects following all trials within an experimental condition. The evaluations were made on five-point rating scales ranging from low to high workload. Several scale items were developed to assess each of the workload dimensions [5].

2.2 Results

2.2.1 Task Time - Performance was approximately twice as fast with the master controller for both tasks. On the task board, the mean task time was 200 and 377 seconds for the master and six-DOF controller, respectively, $F(1,5) = 17.17$, $p < 0.01$. On the ORU scenario, the mean task was 206 and 483 seconds, respectively, $F(1,5) = 7.04$, $p < 0.01$. ORU subtask times were examined to determine whether the large time differences were associated with particular activities such as long translations reflecting longer time to achieve high end effector rates, or fine manipulations (requiring greater control precision). Differences were not found to be associated with particular subtasks.

2.2.2 Manipulator Control Quality - A training effect for error reduction was observed for both control systems (Figure 1). An average error reduction of 2.3 was observed, with the greatest reduction observed between Trials 1-2 and 3-4. By Trial 4 there was no significant difference between the control systems.

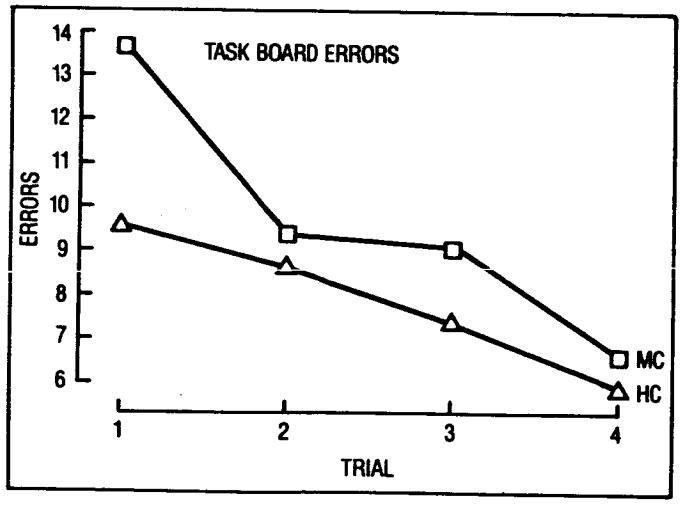


Figure 1. Task board errors as a function of controller type and trial

An additional analysis was made of the relationship between the time it took to complete the task board scenario and the number of errors made. The correlation of errors with time was significant for the hand controller (0.73, $p < 0.01$) and insignificant for the master controller (-0.27). The regression equation for predicting task board time (t) from errors (e) for the hand controllers was: $t = 31.18e + 49.09$. Errors had a major influence on time. Predicted task time with zero errors was 49 seconds. The beta weight for errors indicated that each error increased task time by approximately 31 seconds. The differential relationship between errors and time for the two control systems indicated that different control strategies were utilized.

Few errors were observed on the ORU task; hence, no statistical analysis was conducted. As with the task board scenario, more errors were made with the master controller (seven) than the hand controllers (four).

With respect to ORU control/efficiency ratings, a significant interaction was detected between control system and trial, $F(1,5) = 7.35$, $p < 0.05$. This interaction resulted from a slight drop in average control ratings for the master controller on Trial 2 (from 3.33 to 2.5), while no such drop occurred for the hand controller (mean of 3.33).

2.2.3 Workload - With respect to cognitive workload, three rating scale items were evaluated: attention required, decision making, and skill required. The hand controllers were generally rated higher in workload although the differences were not statistically significant. Based upon subjects' comments, the perception of higher cognitive workload of the hand controllers was based upon: (1) the relatively more complex mental transformations required to position the slave arm end effector, and (2) the absence of more informative force feedback which made decision making in problem situations very difficult.

Three rating scale items were used to evaluate manual control workload: difficulty maneuvering slave arms, difficulty manipulating end effectors, and difficulty controlling parts/equipment. The master controller was generally rated high in manual control difficulty, but again the differences were not statistically significant. Several comments made by subjects indicated that the hand controllers provided superior control in close-in situations where fine, precise movements were required. The master controllers were more difficult to control in that situation, because the force required by the operator to overcome static friction led to relatively quick, erratic movement that did not facilitate control in tight situations. However, irregular and compound axis translations were easier with the master controller.

Physical workload was assessed using three rating scale items: hand/-finger fatigue, arm fatigue, and task related body fatigue. The master controller was rated significantly higher on all comparisons but body fatigue. For hand fatigue the mean ratings were 2.66 and 1.25 for the master and six-DOF controller, respectively, $F(1,5) = 85$, $p < 0.001$. For arm fatigue, the mean ratings were 2.08 and 1.33 for the master and six-DOF controller, respectively, $F(1,5) = 10.29$, $p < 0.01$. Reasons offered by test subjects for greater fatigue of the master included the strength required to overcome static friction and infrequently used muscles.

In the evaluation of overall workload, the master controller was rated as significantly higher in workload due largely to the differences along the physical dimension. The mean ratings were 2.41 and 1.66 for the master and six-DOF controller, respectively, $F(1,5)$ 19.29, < 0.01.

Several interesting comments were offered with regard to overall workload considerations. Concerning the value of force feedback with the master controllers, one subject made the following comment, "I started this task without force feedback and it took very long; I was concentrating so hard on the visual information that I was unaware of the absence of force feedback. When feedback was turned on, there was dramatic difference in presence, realism, and task time." The comment was made after an ORU trial was terminated because the force feedback had not been activated.

3. STUDY 2

In the second study, a truss assembly task representative of the kind anticipated for Space Station was performed. This task was an excellent scenario for evaluating teleoperator control because it required simultaneous operation of both manipulator arms and complex multiaxis slave arm movements.

3.1 Methods

3.1.1 Participants - Six engineers participated in this study. One had extensive experience with master-slave manipulators and a second was familiar with hand controllers. The other four had varying degrees of familiarity with teleoperator manipulator systems, but none were experienced with hands-on operators.

3.1.2 Experimental Design - Two within-subjects factors were investigated: (1) controllers having two levels (master controllers and six-DOF hand controllers); (2) truss strut alignment having three levels (vertical, diagonal, horizontal). Each alignment was thought to represent different problems for the operator. A third factor was investigated in the study, end effector gripper type, but its effect did not impact on the comparison of control systems, so it is not discussed in this paper. Details can be found elsewhere [6].

3.1.3 The Laboratory and Test Equipment - The laboratory and test set-up was essentially the same as for Study 1. Differences included the positioning of cameras at the telerobot and the test article. Cameras were positioned in the center of the slave arms below the shoulder joint and at a 45° angle from the left and right shoulders. The truss structure apparatus was composed of three struts: battens and a diagonal; a truss node mounted to the board on a wooden block, a vertical support board, metal clips attached to the board to retain stowed struts; and three strut-connectors (attached to the node). A sleeve on the strut was moved toward the node and rotated about the strut's longitudinal axis approximately 90° to lock. The orientation of the truss struts was vertical, horizontal, and diagonal with respect to the orientation of the telerobot.

3.1.4 Procedures - Each subject was briefed on the use of the manipulator and was given time to assemble struts for familiarization. During the test, the truss was assembled twice with each control system. The order with which the control systems were used was balanced across subjects. All scenarios were completed with a strut assembly sequence of vertical, diagonal, and horizontal. As in the first study, only CCTV views were used and cameras were voice-command controlled.

3.1.5 Dependent Variables - The dependent variables used in the study were the same as used in Study 1, with one exception: no assessment of errors was made.

3.2 Results

3.2.1 Task Time - Total assembly of all three struts took significantly longer with the hand controllers (1.598 seconds) than with the master controllers (691

seconds), $F(1,5) = 52.82$, $p < 0.001$. An examination of assembly times for individual struts indicated that the master controllers were associated with faster performance for all struts.

3.2.2 Manipulator Control Quality - Test conductor's ratings of the control/efficiency of slave arm movements indicated slightly higher evaluations for the master controllers than for the hand controllers across all three individual strut assemblies, but the difference was not statistically significant.

3.2.3 Workload - Data for all workload variables are presented in Table 1. Cognitive workload was evaluated with the four-scale items presented in Table 1. The hand controllers were rated significantly higher in cognitive workload than the master controllers. Many of the comments offered by test subjects indicated that the hand controllers required much greater mental effort to operate than the master controllers, which were judged to provide more natural control. Since the task required the simultaneous control over two slave arms, the determination of appropriate control inputs was cognitively demanding. This can be readily understood, since the two-arm tasks often required a different input to each controller depending on the orientation of the end effector. In addition, the availability of force reflection in the master arm controllers was an aid to decision making which was absent for the hand controllers.

Manual Control was evaluated using the three-scale items indicated in Table 1. The hand controllers were rated significantly greater in workload for both end effector and equipment control. Comments from the test subjects indicated that the hand controllers were more difficult to use when operating in situations where axes were coupled. This was especially problematic in assembling the diagonal strut, where Y and Z axes were necessarily coupled. With the master controller, these motions were more easily accomplished.

Physical workload was evaluated using the three scale items indicated in Table 1. The master arm controllers were significantly more fatiguing than the hand controllers for arm and hand fatigue, but not for overall body fatigue.

The subjects' assessments of total workload is also provided in Table 1. Interestingly, no significant differences between the control systems was observed. The greater cognitive and manual control workload associated with hand controllers was probably offset by the greater physical workload of the master controllers. This observation illustrates the importance of evaluating individual workload dimensions rather than relying on a single global assessment.

4. CONCLUSIONS

Several consistent patterns were observed across both studies. The time to complete the task scenarios was approximately twice as long with the hand controllers as with the master controllers. The trend toward higher cognitive workload of the cognitive workload of the hand controller observed in Study 1 was confirmed in Study 2. The cognitive complexity of controlling the telerobot with the hand controller became a significant problem when two slave arms were operated simultaneously. Greater physical workload of the master controllers was also found in both studies.

Table 1. Ratings for Workload Variables

VARIABLE	CONTROLLER TYPE MASTER M[1]	MASTER SD[2]	HAND M[1]	HAND SD[2]	F STATISTIC
COGNITIVE					
Attention	3.00	0.63	3.58	0.61	8.45[3]
Decision-Making	2.58	0.45	3.41	0.62	11.36[3]
Manipulator Skill	2.50	0.54	3.75	0.45	25.00[4]
Task Skill	2.75	0.47	3.33	0.51	14.41[3]
MAN. CONTROL					
Slave Arm	1.91	0.70	2.41	0.68	14.42[3]
End Effector	2.00	0.88	3.16	0.69	49.00[4]
Equipment	2.25	0.78	3.41	0.48	2.50
PHYSICAL					
Hand Fatigue	3.00	1.25	1.08	0.24	24.00[4]
Arm Fatigue	2.58	1.09	1.25	0.47	11.03[3]
Overall Fatigue	2.16	1.16	1.41	0.86	3.85
TOTAL WORKLOAD	2.41	1.10	2.83	1.07	1.40

Notes:

1 Average Rating (Higher values = higher workload)
2 Standard Deviation
3 F Statistic is significant at the $p < 0.05$ level
4 F Statistic is significant at the $p < 0.01$ level
Degrees of Freedom = (1,5)

The major advantages of the master replica controllers were their ability to accomplish tasks more quickly and their "naturalness" of control. The latter was especially true when complex multiaxis inputs were required. Their major disadvantages were less control over sustained movements requiring very fine, precise control, and high operator physical workload/fatigue. The major advantages of the hand controllers were the control that could be achieved over precision movements, and very low physical workload. The major disadvantage of hand controllers was the cognitive complexity of correlating control inputs with slave arm outputs, especially when the operator was coordinating movements of two manipulator arms with the end effectors in different orientations. Further, no simple solution can be implemented, such as altering the coordinate reference frame, as is done with single manipulator arm systems, or using an

end-effector mode similar to that of the Shuttle RMS. With a two-arm telerobot, each arm would have an independent end-effector mode, which would require two separate camera views. This would make simultaneous control of both arms difficult.

Several improvements to the present hand controller design and implementation are being studied. First, the auditory-force feedback was not sufficiently informative to the operators. Providing force reflection to the controller itself, or a force/torque graphic display is being investigated. Second, alternative rate and position control options will be made available to the operator. Third, alternative controller displacement-rate functions such as variable slope (providing a fine to course gradient), exponential, or linear-linear "dog-leg" are being investigated.

5. ACKNOWLEDGEMENT

Portions of this work were supported by NASA, contract (NAS 9-17229). Appreciation is expressed to Lyle Jenkins, NASA Technical Monitor, for his assistance. The opinions expressed are those of the authors alone and not necessarily NASA, or the Grumman Corporation.

6. REFERENCES

[1] Rice, J., Yorchak, J., and Heartly, C., "Planning for Unanticipated Satellite Servicing Teleoperations," Proceedings of the Human Factors Society 30th Annual Meeting, pp. 870-874, Dayton, OH: Human Factors Society, 1986.

[2] Brooks, T. and Bejczy, A., Hand Controllers for Teleoperation, (JPL Publication 85-11), Pasadena, CA: Jet Propulsion Laboratory, 1985.

[3] Heartly, C., Cwynar, D., Garcia, K., and Schein, R., "Capture of Satellites Having Rotational Motion," Proceedings of the Human Factors Society 30th Annual Meeting, pp. 875-679, Dayton, OH: Human Factors Society, 1986.

[4] O'Hara, J., and Olsen, R., "Control Device Effects on Telerobotic Manipulator Operations," Robotics, Volume 4, pp. 5-18, 1988.

[5] O'Hara, J., "The Effect of Teleoperator Control Methodology on Task and System Performance Using a Remote Dextrous Manipulator, Master/Replica Controllers Versus Six Degree-of-Freedom Hand Controllers," Report No. 5A-MSET-FR-8601, Bethpage, NY: Grumman Space Systems, 1986.

[6] O'Hara, J., "Space Station Truss Structure Assembly Using a Two-Arm Dextrous Manipulator with Varying Control Systems and One Effector Design," Report No. SA-TWS-86-R007, Bethpage, NY: Grumman Space Systems, 1986.

Autonomous Sensor-Based Dual-Arm Satellite Grappling

Brian Wilcox Kam Tso Todd Litwin Samad Hayati Bruce Bon

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

The NASA Telerobotic Research Project is exploring the feasibility of using robots in space for on-orbit assembly, maintenance, and repair operations. Dual-arm satellite grappling is one of its more challenging tasks.

The task involves the integration of technologies developed in the Sensing and Perception (S&P) Subsystem for object acquisition and tracking, and the Manipulator Control and Mechanization (MCM) Subsystem for dual-arm control. S&P acquires and tracks the position, orientation, velocity, and angular velocity of a slowly spinning satellite, and sends tracking data to the MCM subsystem. MCM grapples the satellite and brings it to rest, controlling the arms so that no excessive forces or torques are exerted on the satellite or arms.

The demonstration setup includes a 350-pound satellite mockup which can spin freely on a gimbal for several minutes, closely simulating the dynamics of a real satellite. The satellite mockup is fitted with a panel under which may be mounted various elements such as line replacement modules and electrical connectors that will be used to demonstrate servicing tasks once the satellite is docked.

The subsystems are housed in three MicroVAX II microcomputers. The hardware of the S&P Subsystem includes CCD cameras, video digitizers, frame buffers, IMFEX (a custom pipelined video processor), a time-code generator with millisecond precision, and a MicroVAX II computer. Its software is written in Pascal and is based on a locally written vision software library. The hardware of the MCM Subsystem includes PUMA 560 robot arms, Lord force/torque sensors, two MicroVAX II computers, and Unimation pneumatic parallel grippers. Its software is written in C, and is based on a robot language called RCCL.

This paper describes the two subsystems and provides test results on the grappling of the satellite mockup with rotational rates of up to 2 rpm.

1 Introduction

NASA is exploring the possibilities of using autonomous systems in place of astronauts for dangerous or time-consuming operations, such as extra-vehicular activity. The NASA Telerobot Research Project is looking at the use of autonomous systems for performing on-orbit assembly, maintenance, and repair operations. All of these operations present challenges to the field of robotics.

This paper is concerned with one challenge associated with the repair of artificial earth-orbiting satellites. That challenge centers around the fact that most such satellites are spin-stabilized. In

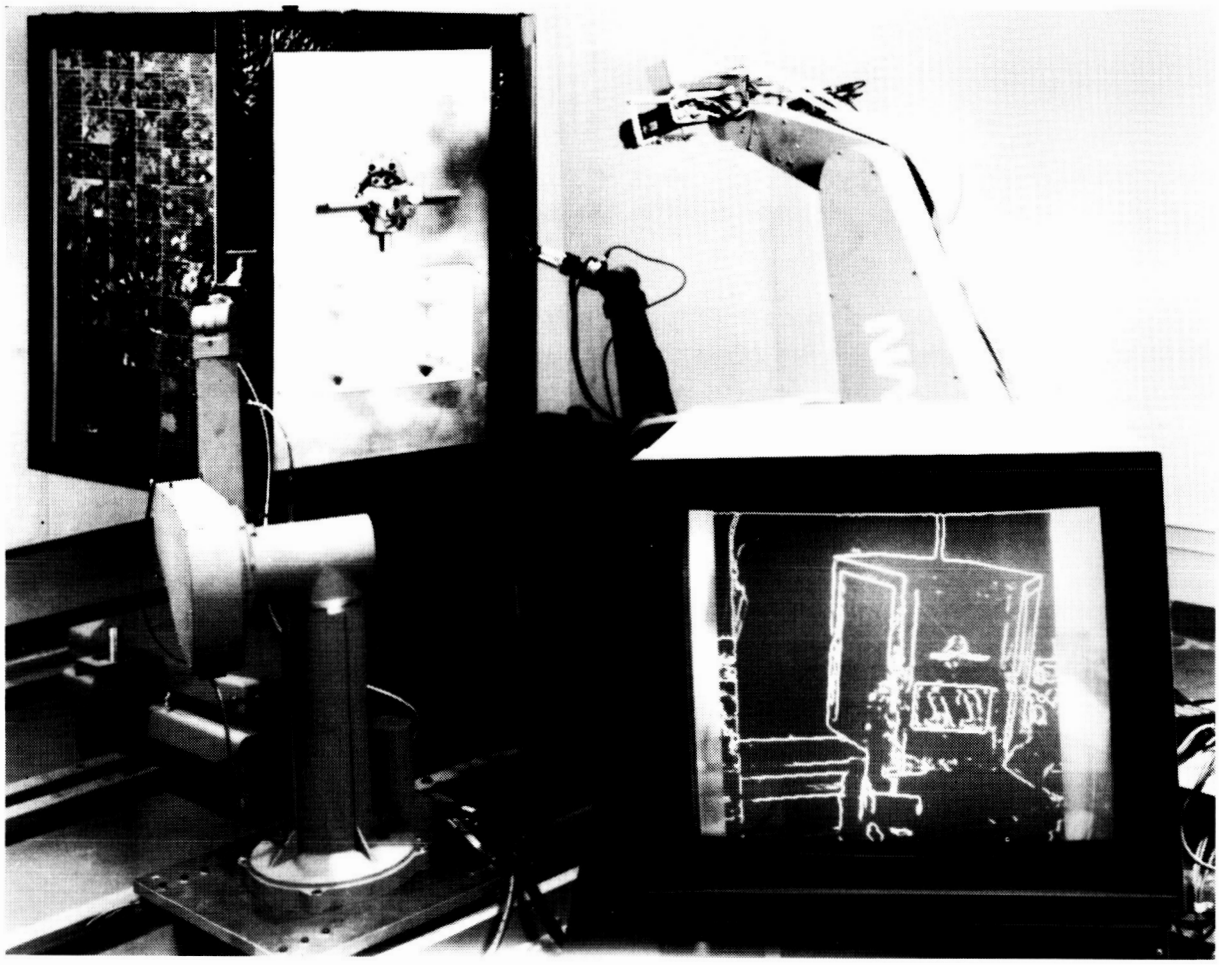


Figure 1: Testbed Setup

order to repair such a satellite, it must first be de-spun. An investigation was made into the use of a dual-arm robot with visual assist to grapple and de-spin a rotating satellite mockup.

The Testbed of the NASA Telerobot Project, housed at the Jet Propulsion Laboratory, is composed of several subsystems. Two of these subsystems, the Sensing and Perception (S&P) Subsystem and the Manipulators and Control Mechanization (MCM) Subsystem, were directly involved in the effort to grapple a spinning satellite. S&P visually acquired and tracked the rotating satellite and transmitted the satellite's state information to MCM in real time. MCM, using the data from S&P, directed the arms to grab the satellite and to stop its motion as shown in Figure 1.

Below we will discuss these two subsystems and provide the details of the satellite grappling technique.

2 Testbed Setup

The Testbed of the NASA Telerobot Project is a facility which is composed of many parts. It is divided into three main sections: the computer facility, the operator control station, and the test area.

In the computer facility there are various computer systems to support the subsystems. Included among them are several MicroVAX II microcomputers, Sun workstations, and a Symbolics Lisp machine. There are also other specialized processors used at low levels of support.

The operator control station provides a place for the operators during Testbed use. It has working surfaces, computer terminals, video monitors, and joysticks. It is situated so as to give a good view of the test area.

The test area contains everything else. It houses the two manipulation arms, plus a third arm used for holding movable video cameras (not used for satellite grappling), and all of the video cameras. The satellite mockup, as well as other targets for telerobotic research, are in this area. In addition, there is a calibration fixture used to calibrate the cameras in S&P and the arms in MCM to a common coordinate system.

Satellite Mockup

The satellite mockup is a 350-pound model of a generic artificial satellite. It is suspended from a counterweight by a long cable which is connected — through a fairly wide opening in the top of the satellite — to a gimbal near its center of mass. The method of suspension allows the satellite to move within a small area in a manner similar to its free-space counterparts.

The sides of the mockup consist of real solar panels, framed in the gold-foil insulating material typical of real satellites. This gives S&P a realistic visual target, complete with visual clutter and specular reflections.

On one side of the mockup there is a removable task panel, under which may be mounted various elements such as line replacement modules and electrical connectors. These can be used to demonstrate servicing tasks once the satellite is grappled and docked.

S&P Hardware

The S&P Subsystem hardware consists of a DEC MicroVAX II microcomputer and several other pieces of special equipment. There is a time-code generator, shared with MCM, which allows time tagging the data to millisecond precision. There are five video cameras. Three of them are mounted to the rear of the test area, yielding good overall views of the work region. The other two are mounted on a robot arm and are used for close-up views; they were not used in the work described here.

The cameras feed their signals into video digitizers, which in turn feed a custom video pipelined processor. The processor is called IMFEX, the Image Feature Extractor, and it uses a thresholded modified-Sobel operator to find high-contrast edges in video images. The output of IMFEX is fed into video buffers for storage, access, and manipulation by the MicroVAX II. The video buffers have internal D/A converters to allow viewing of their contents, which may be a captured frame, graphics generated by the MicroVAX II, or a combination of the two. A 16-by-16 video crossbar

switch is used to route all of the analog video signals between the components mentioned above, and over to the video monitors in the operator control station for display.

S&P Software

The S&P software is composed of two major sections, both written in VAX Pascal. The first section is a 30,000-line general-purpose vision software support library. It contains the software used to control and access the various hardware components of S&P, as well as routines to perform all manner of support services for machine vision. Included in this library are the routines used to perform the functions of acquisition and tracking described below, to perform camera calibration, and to deal with object models. The second major section of the software is a 20,000-line software package which is the S&P-specific application code. It is decomposed into five separate tasks, all running concurrently.

MCM Hardware

The MCM Subsystem hardware consists of two PUMA 560 robot arms. Each arm is equipped with a Unimation pneumatic parallel gripper which has been fitted with simple parallel fingers holding a special grappling tool. Each tool has a flexible handle and a Velcro pad at the end in order to hold to the satellite mockup, which has the two complementary Velcro pads attached on either side of the task panel. Each arm is also equipped with a commercial (Lord Corporation) wrist force/torque sensor. The wrist force/torque sensors are used to read the forces and torques sensed in all six dimensions. The computing hardware consists of two DEC MicroVAX II computers. Each one is interfaced to the LSI 11/73 microprocessors of the Unimate controller via two DRV11 parallel cards. The Unimate controller reads the LORD force/torque sensor data through another DRV11 parallel card. The S&P object state data is obtained through a 9600 baud RS-232 serial line connected between the S&P MicroVAX to the MCM MicroVAXs. Figure 2 shows a schematic drawing of the hardware.

MCM Software

The MCM MicroVAXs run under a modified Berkeley BSD 4.3 Unix operating system. The main robot language is RCCL (Robot Control C Library) which was developed originally at Purdue University by Vincent Hayward [3] based on Richard Paul's robotics textbook [4] and later enhanced by John Lloyd [5] and ported to a MicroVAX II [6]. RCCL is a collection of C functions which provide easy and general robot programming. The software can be partitioned into two main parts: the planning level and the real-time control level.

The planning level consists of functions that allow the programmer to specify desired coordinate frames for the robot end-point target position. The programmer must specify several frames, such as where the robot is in the world frame and in the tool frame. He/she must also specify the velocity, and whether the motion should be carried out in Cartesian or joint-interpolated modes. The relationships between the various frames are entered in the code exactly as one would write them in mathematical notations. A function called Makeposition interprets the code and sets up

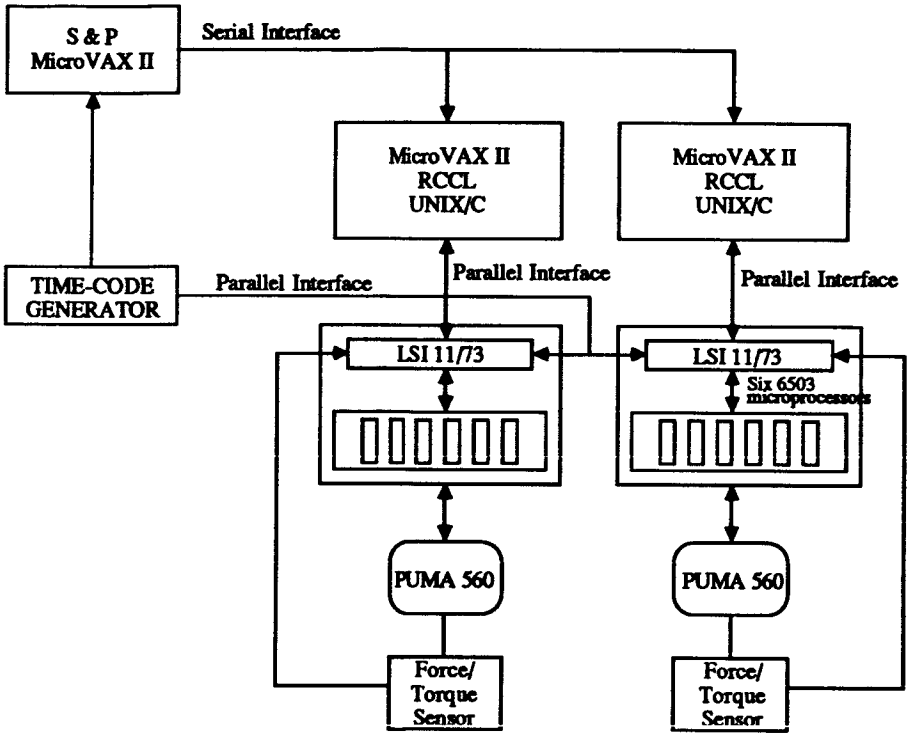


Figure 2: Schematic drawing of the hardware

appropriate matrix equations. Its general form is:

$$\text{p} = \text{Makeposition}\ (\text{“P”}, Z, \ldots, T6, \ldots, R, EQ, A, B, \ldots, U, \text{TOOL}, R)$$

This simply sets up a matrix equation

$$(Z\ldots)T6(\ldots R) = AB\ldots U \tag{1}$$

The main objective is to solve this equation for T6 as

$$T6 = (Z\ldots)^{-1}(AB\ldots U)(\ldots R)^{-1} \tag{2}$$

where $T6$ represents a homogeneous transformation relating the link 6 frame in the 0th (or shoulder) frame. The planning level runs in a normal time-shared manner. A program can develop many motions and place them in a queue for the control level to execute sequentially. An important feature of RCCL is that it allows one to modify the matrices in equation (1) in the control level.

These modifications can be triggered either by the planning level or by external sources such as from S&P object state data or force/torque data. This feature has been used extensively in our work to implement both the tracking and force/compliant control during grappling the satellite mockup.

The control level which is called RCI (Robot Control Interface) is a general robot programming environment. One can write his/her own robot control programs and create custom trajectories to run the robots. In the normal RCCL operations this level receives the motions from the motion queue and uses a trajectory generator to interpolate between the specified via points. Finally, inverse kinematics is performed to obtain the joint angle set points to be sent to the joint microprocessors via the LSI 11/73 computer.

The system works by a hardware clock (located in the Unimation controller) which periodically sends an interrupt to a communication program in the LSI 11/73. At every interrupt, this program gathers information from the arm — which includes the joint angles and other sensors that have been interfaced to the LSI 11/73 — and makes them available to the MicroVAX in shared memory; it signals the MicroVAX with a hardware interrupt. The MicroVAX reads these data and immediately sends the joint angles that have been computed in the previous cycle back to the LSI 11/73 for execution. Note that in this implementation the VAL language is completely bypassed. The hardware of the Unimate controller remains intact, however, and one can switch between the VAL language and RCCL without any hardware modifications.

3 S&P Acquisition and Tracking Algorithms

The task of watching a moving object is broken down into two stages, the first of which is called acquisition. This is the stage wherein the object of interest is first localized within the views of the cameras, and an initial computation is made as to its location and movement within 3-space. It is computationally intensive, and cannot perform quickly enough on currently available computers to keep up with a moving object in real time.

The second stage is call tracking. Tracking is more computationally efficient than acquisition, and is used to follow the object as it moves, updating the state information that was initially provided by acquisition.

Both stages compute the following time-tagged state information in three dimensions: position, translational velocity, orientation, angular velocity, and a covariance matrix of these values.

Acquisition

A fully autonomous acquisition algorithm is currently under development at the Jet Propulsion Laboratory and was not tested in the grappling experiment. A moment's reflection, however, will reveal that it is not possible to track an object without first acquiring it in some fashion. In order to satisfy this need, a "quick-and-dirty" operator-assisted version called *hand acquisition* was designed for the current work.

In hand acquisition, an *a priori* position is used as a starting point. Using this position, S&P displays a wire-frame projection of the satellite's object model in the display, superimposed on the raw video. While the satellite was held still, the operator uses the joysticks to move the wire-frame overlay — and thus the state of the object model — until it roughly overlaps the satellite mockup

in all camera views. Once complete, the operator signals that tracking may start. Hand acquisition of a moving satellite was attempted with mixed results.

Tracking

The tracking algorithm was designed by Donald B. Gennery. Detailed descriptions of the algorithm are given elsewhere [1,2]. The tracker performs its operations in five major phases. In the first phase it acquires a frame of video and notes the time tag associated with the data. In the second phase the old object state is propagated forward to the time of the new data. In the third phase a projection of the propagated object state's edges is made into the view of the camera which took the new data. In the fourth stage measurements are made of the discrepancies between the locations of edge points in the projected edges and the locations of edge points in the data. In the fifth and final stage the projected object state is adjusted by using a least-squares technique with respect to the measurements taken in the fourth stage. Uncertainties are propagated and determine the effect that any given data set has on the current object state.

These five stages constitute one tracking cycle. Between cycles, the updated state information is sent to MCM. Then the tracker selects the next camera and performs another tracking cycle. It continues in this fashion until told to stop or until it loses track. If track is lost, S&P cycles back into acquisition and repeats the entire process.

4 MCM Tracking and Grappling Algorithms

The two robots are driven independently by two MicroVAXs. They run the same copy of the software except each has its own coordinate frames because of the different locations of the robots and because they grapple different points on the satellite mockup. Two machines are used because the computing power of one MicroVAX is not adequate to control two robots during the tracking phase. The two robots are coordinated because they are basically driven from the same source of data — the S&P object states of the satellite mockup.

The MCM software receives the satellite object state at a rate of about two times per second. The object state consists of a time-stamp, position vector and orientation quaternion of the centroid of the satellite mockup, the translational and angular velocities, and their covariances.

A complete cycle of satellite grappling is comprised of four phases: *approach*, *tracking*, *grappling*, and *docking*.

In the approach phase, the MCM software monitors the orientation and angular velocity of the satellite mockup. It deploys the robots to the approach positions when the satellite mockup is spinning with a rate at or below two rpm. The approach positions are chosen so that the robots have the maximum work space for tracking and grappling. The approach position of the left robot is described by the follow equation:

$$ {}^{w}_{0}T_{L}\ {}^{0}_{6}T_{L}\ {}^{6}_{e}T_{L} = {}^{w}_{a}T_{L} \tag{3} $$

where ${}^{w}_{0}T_{L}$ is a transformation describing the 0th frame, ${}^{6}_{e}T_{L}$ is the tool frame, and ${}^{w}_{a}T_{L}$ is the approach frame of the left robot as shown in Figure 3.

The robots wait in the approach positions until the mockup has rotated such that the pads are facing the tools with a designed distance of 100mm. At that moment the robots start tracking the

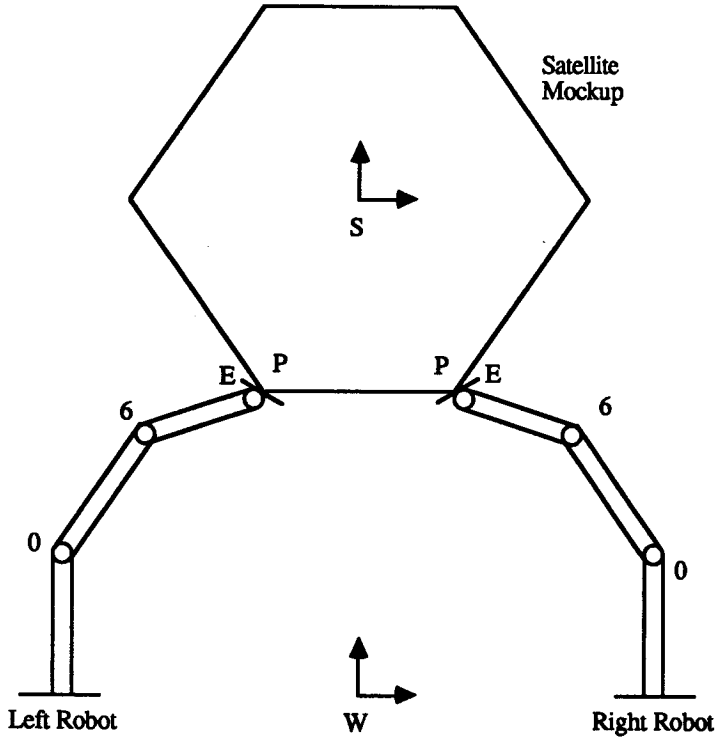


Figure 3: Coordinate Frame Assignments

satellite mockup driven by the S&P object state data. At the same time the distances between the tools and the pads are gradually reduced until they contact each other. The following kinematic equation is used to specify the motion in the tracking phase:

$$ {}^{w}_{0}T_L \; {}^{0}_{6}T_L \; {}^{6}_{e}T_L = {}^{w}_{s}T \; {}^{s}_{p}T_L \tag{4} $$

where ${}^{w}_{s}T$ is a transformation describing the centroid of the satellite mockup frame, and ${}^{s}_{p}T_L$ is the transformation from the satellite mockup centroid to the left pad. The distance between the tools and the pads, initially 100mm, is faked in equation (4) by making the tool 100mm longer than its physical length. This distance is reduced during tracking until contact.

In RCCL, one can generate trajectories by using the trajectory generator or by an external means. The latter is made possible since one can modify any of the transformations except $T6$ in equation (1) in real time. The satellite tracking uses the latter strategy since it would be too time-consuming if planning is done each time MCM receives an updated object state.

Hence tracking is done in the control level which drives the robots in the following way: Every time an object state is received, the current frame of the satellite ${}^{w}_{s}T$ is predicted according to the received data, and the frame of the tool ${}^{w}_{e}T$ is computed from the joint angles of the robot. The

difference between ${}^{w}_{p}T_L$ and ${}^{e}_{w}T_L$ is computed from

$$\Delta_L = {}^{e}_{p}T_L = {}^{e}_{w}T_L \ {}^{w}_{p}T_L \tag{5}$$

where ${}^{w}_{p}T = {}^{w}_{s}T \times {}^{s}_{p}T$. In order to track the satellite, the robots are required to have moved this Δ by the time the next state update is received. This means that, in every sampling period, the robots are moved by $\delta = \Delta \times \frac{t}{T}$, where t is the robot control sampling period, and T is the inter-arrival period of the object state.

The tools approach the grappling pads until a contact is initiated. This is sensed by the force/torque sensors and after the contact the motion is changed from vision-based servoing to force servoing. The following kinematic equation is used to specify the motion in the grappling phase:

$${}^{w}_{0}T_L \ {}^{0}_{6}T_L \ {}^{6}_{e}T_L = {}^{w}_{s}T \ {}^{s}_{p}T_L \ \text{COMPLY} \tag{6}$$

where COMPLY is a "small" time-varying transformation and has the following form

$$\text{COMPLY} = \begin{bmatrix} 1 & -\delta\theta_z & \delta\theta_y & \delta x \\ \delta\theta_z & 1 & -\delta\theta_x & \delta y \\ -\delta\theta_y & \delta\theta_x & 1 & \delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Since the COMPLY transformation is placed after the ${}^{s}_{p}T_L$, it will modify the ideal trajectory by a small amount each sample time. Because of integral force control, the COMPLY transformation will be modified based on the force/torque sensor readings and will keep its value even after the forces have been nulled. Since the satellite mockup cannot be stopped instantaneously once it is grappled, the software decelerates the mockup according to a trajectory which is generated based on the initial velocity at the moment of contact.

Once the satellite mockup is stopped, it is pulled to the docking fixture. Active force control is used to nullify the force built up due the dual-arm coordinated motion. If grappling does not occur — detected by the lack of contact between the tools and pads — the whole cycle is repeated by letting the satellite mockup spin one more time.

5 Conclusions and Future Improvements

We have successfully grappled the satellite mockup with rotational rates of up to 2 rpm. With higher speed, due to the communication delay and control inaccuracies, the robots start to miss the pads.

In the present MCM implementation, the computation is performed with two MicroVAX computers, which limits its control rate to once every 28 msec. In the near future, we plan to port our software to a Sun 4/260 computer which will increase the control rate to 200 Hz. This increase will improve the force control capability and hence reduce the build-up of forces at the moment of contact and subsequent grappling. The improved version of RCCL [7,8] can coordinate trajectories for two robots. As such, more precise coordination both at the planning and control levels can be achieved.

6 Acknowledgements

The research described in this document was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] D. B. Gennery, "Tracking Known Three-Dimensional Objects," *Proceedings of the AAAI Second National Conference on Artificial Intelligence*, Pittsburgh PA, August 1982, pp. 13-17.

[2] B. Wilcox, D. B. Gennery, B. Bon, and T. Litwin, "Real-Time Model-Based Vision System for Object Acquisition and Tracking," *Proceedings of the SPIE International Conference*, Los Angeles CA, January 1987.

[3] V. Hayward, and R. Paul, "Robot Manipulator Control Under Unix RCCL," *The International Journal of Robotics Research*, Vol. 5, No. 4, Winter 1987, pp. 94-111.

[4] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, 1981.

[5] J. Lloyd, "Implementation of a Robot Control Development Environment," M.S. Thesis, Department of Electrical Engineering, McGill University, Montréal, Québec, 1985.

[6] J. Lee, S. Hayati, *et al.*, "Implementation of RCCL, a Robot Control C Library, on a MicroVAX II," *Proceedings of the SPIE Conference on Advances in Intelligent Robotics Systems*, Vol. 726, October 1986, Cambridge MA, pp 26-31.

[7] J. Lloyd, M. Parker, and R. McClain, "Extending the RCCL Programming Environment to Multiple Robots and Processors," *Proceedings of the IEEE International Conference on Robotics and Automation*, Philadelphia PA, April 1988, pp. 465-469.

[8] S. Hayati, T. Lee, K. Tso, P. Backes, and E. Kan, "The JPL Telerobot Manipulator Control and Mechanization Subsystem (MCM)," *Proceedings of the NASA Conference on Space Telerobotics*, Pasadena CA, January 1989.

Thread: A Programming Environment For Interactive Planning-level Robotics Applications

John J. Beahan, Jr.[1]

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

This paper discusses the Thread programming language, which was developed to meet the needs of researchers in developing robotics applications that perform such tasks as grasp and trajectory design and analyzing sensor data, and which interface with external subsystems in order to perform servo-level control of manipulators and real-time sensing. This paper discusses the philosophy behind Thread, the issues which entered into its design, and covers the features of the language from the viewwpoint of both researchers who want to develop algorithms in a simulation environment, and those who want to implement physical robotics systems. This paper does not attempt to explain the detailed functions of the many complex robotics algorithms and tools which are part of the language, but only to give an impression of their overall capability.

I. Introduction

This paper mixes many concepts from robotics, programming language design, large-scale software engineering and systems theory. Because of the unusual set of issues addressed, and the fact that some of them may be rather esoteric to some members of the robotics community, this paper mixes abstractions and "nuts and bolts" examples rather freely, with little middle ground, which will hopefully not prove too disjointed.

The paper will first explain the philosophy behind this work, then cover a bit of the background, rationale and goals it tries to address. The Thread language is then described, first from the viewpoint of software engineering, then from the viewpoint of the specific robotics capabilities it provides. The software features are covered first because they contribute greatly to the utility of the language in developing algorithms and performing experimental robotics, and have a significant impact in how the customized robotics facilities can be used. The language's individual features will first be discussed in an abstract manner, then with specific examples, from the viewpoint of both robotics and software engineering. A concluding section will discuss the lessons, both positive and negative, which were learned during the development and use of the language.

Thread is an interpreted (threaded) language, implemented in Ada, which provides an environment for interactive program development using a wide variety of language features specialized for robotics. The language was developed to meet the needs of the Run Time Control (RTC) subsystem of the JPL Telerobot Demonstrator, which is aimed at performing flexible satellite servicing in space using a cooperative man-machine system partitioned into a base site (ground or space station/shuttle) and a remote site. The Telerobot system is organized hierarchically by bandwidth of contact with the physical manipulator hardware, with an interactive Operator Control Station connected to a Task Planning and Reasoning subsystem, which then sends the RTC goals and constraints expressed as English-like commands such as "`move the right arm to a location 10.0 cm above the Door_handle top using elbow_up poses please.`". The RTC performs the detailed numerical planning steps and sensor monitoring operations necessary to execute the command and recover from simple anomalous events, calling on a manipulation subsystem and a machine

[1]This work was performed with the generous support of Guillermo Rodriguez, Supervisor for the JPL Tele-Autonomous System Research Group.

vision subsystem for functions such as joint and cartesian motion, compliant hinge rotation, intelligent "macros" such as bolt threading and 3-d real-time object position determination. The Telerobot system is very large, consisting of a network of five primary computers and roughly 200,000 lines of embedded software, which use a highly diverse mix of operating systems, programming languages and implementation philosophies.

Philosophy

The philosophical starting points for the development of Thread were ideas of object-oriented design from computer science, integration and robustness issues from software engineering, coordination and dependency concepts from systems theory, and abstraction principles from programming language theory.

The number of additional issues which computer implementation adds to the problems of robotics is such that it is not unusual for the implementation details of hardware/software development to dominate all other concerns in time and effort. The magnitude of the software overheads can be significantly lowered, however, by using advanced software engineering methods, and techniques from some of the less well-known branches of computer science. Thread is an attempt to reduce some of the many overheads associated with implementing robotics concepts in software, and this paper tries to convey the philosophy and methods which went into the design of Thread. The intention of this paper is not merely to recommend Thread as an environment in which to build general robotic systems, although it is a very powerful tool for the class of problems it was designed to address, but rather to explain the ideas behind Thread so that they may be applied to implementing other robotic systems in whatever particular context is of interest.

The system which Thread was developed to serve, the RTC, was a large project involving many developers, and this led to the fact that the large-scale system issues and software engineering concerns were likely to dominate development unless significant investments were made to manage them. A strong commitment was made from the outset to accept the additional initial investment necessary to insure proper flexibility and generality in the future. Also, the needs of the RTC necessitated that the language be extensible at the compiled level, extremely interactive, and be able to be made extremely robust to runtime errors. For these reasons, Ada was chosen as the implementation language and the decision was made to use Thread as the RTC development environment. In addition, the observations made during many large system development efforts [3] to the effect that peripheral concerns with implementation details can make up the majority of the development effort, sometimes over 90%, led to the need to reduce from the outset the number of minor software implementation details the RTC developers must deal with, and allow them to work in close proximity to the actual issues of robotics, rather than being distracted by software issues.

There were also several other issues associated with the development of large systems which were prominent in the original conception of the Thread language. A well-known result from software engineering studies is that productivity, in terms of quantity of software per unit time, is relatively independent of the level of abstraction in the language used, i.e., writing in a low-level language with simple instructions is neither significantly faster nor slower than writing in a highly abstract one with very powerful individual instructions. Therefore, unless computational efficiency prevents it, one should attempt to use as abstract a language as possible, to achieve maximum functionality with as little software as possible.

Another well-known aphorism of software engineering is that "the production of a large system that works is practical only by extension from a small system that works", by building in layers upon a firm foundation. It is utterly impractical to build a deep multilayered system if the lower levels are not initially built to be very general and robust so that the later work can be productive, rather than being largely devoted to revisiting and redressing the design of the lower levels [3]. When an implementation is built to solve a problem, it should solve that problem once and for all, and be able to be reused *with no significant change or effort* in whatever context that problem arises. Implementations which do not generalize are a complete waste of time for large systems, which are long-term and diverse enough that specialized implementations will end up being rebuilt several times in slightly different ways, at roughly the same cost each time.

For similar reasons, it is also unacceptable that the products of several separate research and development efforts should require more than moderate effort to be integrated. Much of the implementation of robotics research systems is carried out in a manner which isolates the results of different researchers from each other, so that even though one research group may have a very powerful manipulator control system, capable of sophisticated force control, and another may have a very flexible kinematics/dynamics simulation environment and graphics interface, integrating these two at anything more than a superficial level usually requires that the two software (and sometimes the hardware) systems be extensively modified, if not virtually rebuilt from scratch. Although this situation is thought to be a fact of life by many people, methods do in fact exist to guide the implementation of systems so that they are very robust to shifting requirements and much more able to accommodate changes. This not only allows separately developed software to be more easily integrated, but more importantly, it leads to the development of software which is far more generally applicable, and thus allows the development effort to easily and safely build progressively higher on previous work.

Related Work

There have been a number of languages specialized for robotics [9,18,10], but virtually all of them have been intended for use as standalone systems, not as components in a very large integrated system. For that reason, most of the requirements placed on Thread were not met, and were often not desirable for previously existing robot languages. Many were overlayed on existing low-level languages [10,18] and thus, although they do provide specialized facilities for robotics, do not relieve the user of any of the minutiae of software development. Also, systems have traditionally been much smaller, and it has been only recently that a robot system could afford to rely on separate subsystems to perform servo control and real-time sensing and concentrate on the planning-level issues which arise when flexibility becomes the primary goal; previous robot environments have been devoted primarily to the performance of very rigid tasks in an extremely structured environment.

The characteristics of the Thread language actually place it closer to one of the modern very-high-level specialized languages such as *Mathematica*™, than to any traditional robot control language, as it has a much stronger interactive orientation than do traditional automated robotics systems.

Rationale and Goals

The rationale behind the design of Thread involved a number of factors, the first of which is the incremental cycle of design and implementation inherent in a research environment, which requires from the outset that the system be able to cope easily with significant design changes which occur as experience and understanding is gained from initial prototypes. This leads to a heavy emphasis on flexibility rather than performance, reinforced by the fact that the system is intended as a rapid-prototyping research tool rather than an application system. Also, there are typically many more developers and users of software at the planning level than at the servo level of robotics, and this also leads to a decreased emphasis on optimally efficient implementation and more concern with issues of flexibility, reliability and abstraction than has historically been present in most robotics implementations. The large size of the Telerobot led to the RTC development team numbering five people, which created much stronger requirements for formal software development discipline than is typical in a research project. The tight integration between the various subsystems in the Telerobot meant that faults and liabilities in one subsystem could quickly collapse the system as a whole if they were not carefully controlled, putting heavy burdens of reliability on the software. The fact that software behavior was strongly dependent on physical interactions in the environment led to the need for a completely interactive development environment which could be easily and safely used to perform software development during laboratory experimentation with actual manipulators. The strongest contribution to the overall design requirements of the language, however, was the desire to support an extensible "tool-oriented" development environment, which would give the developer ***and the user*** the freedom to use the system either as a single program, providing complete robotic tasks as inputs, or in a very piecemeal way, using the various functional modules of the system individually, to control the precise details of a portion of a task.

Tool-oriented Philosophy

The desire is for the Telerobot to be usable at either a very high level, or to have the operator intervene and aid in some operations, at whatever level of detail is desired or appropriate. The implementation structure chosen for the Run Time Controller is a layered set of mutually supportive tools, with the highest level made up of a few large, separate modules, used as utilities with conventional menu/graphics interfaces, for planning obstacle-free paths, generating trajectories, simulating grasp effectiveness, modifying the world model, executing actuation primitives which physically move arms, accessing the vision system, etc. These large modules are built up from toolkits designed for building path planners, trajectory generators, etc., *each with its own appropriate graphical/ textual interfaces.* These tools are built out of smaller, simpler tools, corresponding roughly to conventional software utility libraries, but *usable in an interactive manner.* The lowest level is simple numeric operations, control structures, etc. plus customized robotics features, *all interactively available at any time.*

The benefits of this method are myriad, including: (i) as mentioned above, the operator naturally has visibility and access into all points in the structure of the system at all times; (ii) the design of a tool-oriented implementation will probably be far superior to a monolithic design in generality and robustness, since far more thought and effort will have gone into organizing its subtleties and interrelationships into a coherent structure; (iii) because tools are far more easily reusable than specific programs, it allows construction of later work by relying heavily upon what has already been built, in a much more efficient way than more specialized development styles, which often repeatedly replicate effort.

The drawbacks of this implementation style lie mainly in the complexity of the resulting interfaces, which typically are inefficient at hiding detail from the user; overlaying the textual interface with a graphical/ iconic one is quite feasible, and may go a long way toward alleviating this problem.

II. The Thread Language

Overview

Thread's set of syntactical and semantic features are somewhat unusual, but most of its characteristics are found spread among several other existing languages. Thread allows the use of several different formalisms of programming, including applicative and object-oriented programming along with the conventional style, and the language supports the standard complement of basic data types, operators, control structures, file access facilities, access to the operating system, and various utilities such as a help system and a dictionary of user-defined objects and procedures. The language's syntax superficially resembles Forth [4], but the semantics are closer to that of very abstract languages such as Smalltalk [7,8] or LISP [19]. Thread's design principles are an attempt to apply very abstract programming language concepts to make *building tools* as convenient as possible, rather than focusing only on simplifying the task of writing applications programs.

The most unusual aspect of the language is that it has no input primitives of any kind: the only method of input into a program is the interpreter itself, used recursively. Because of the simple syntax of the language and the interpreted nature of its implementation, any code fragment in any form behaves as an entire program, able to be compiled to the internal form used by the interpreter and executed. There is thus no distinction made betweeen data and code, which naturally gives the ability to treat user input, or the contents of a file or text string, as if it were an executable function. This is a very powerful feature, which can be used in a variety of useful, if slightly unorthodox ways, as will be discussed later.

The fundamental design concept of the language was extensibility, to allow the language to be extended with new routines by entering interpreted code, and also to allow the modular addition of new primitive functions, in the form of compiled code, into the language interpreter kernel itself. An extremely simple interface exists between the Thread language and several compiled languages (Ada, C, Fortran, Pascal) which allows the user to use existing high-speed software developed in conventional environments to add new primitive operators to the language, usually at a cost of only a few lines of interfacing "wrapper". The dual compiled/interpreted nature of the language allows the use of compiled code for high-speed or well-established functions, and interpreted code for rapid prototyping and fluctuating components which are used directly by the human operator. Also, because of the extreme modularity of the implementation of the language kernel, functions are easily extensible and shiftable between compiled and interpreted levels with little overhead.

Syntax and Semantics

The syntax is a simple postfix notation, read from left to right, with each data item implicitly pushed to a global stack and each procedure executed by taking its arguments from and leaving its results on the stack. This has the effect of making every single function and subroutine automatically into a facility which can be directly accessed from the keyboard, which is a strong encouragement to construct programs so that they can more conveniently and modularly be tested and debugged, thereby encouraging generalized tool building rather than specialization. No distinction is made by the language between language primitives and user-defined procedures; this encourages the development of small, modular, carefully tested units of software which can be shared among many developers with a minimum of overhead. (Typically, the only documentation needed for the new user of the language is a short, hands-on tutorial and a list of the names and argument formats of the available primitives and user-defined procedures.) This stack-oriented notation is prevented from becoming cumbersome because procedures possess local stack contexts, use named arguments and named local variables to store data. This removes the heavy burden of stack manipulation which is common to other stack-oriented languages.

The following figure gives a very quick overview of the basic syntax and semantics of Thread, which is provided as an introduction to the many short examples given later. (The convention for naming objects in Thread is that data items begin with an uppercase letter, and procedures are lowercase.)

```
"Hello, world." print.                          A program which prints a string.

2 3 plus 4 times print.                         Calculates the number 20.

3.2 make A_variable.                            Create a variable.

"Something else." store A_variable.             Store something else into it.

1 define factorial                              Defining a simple function.
        N
    if N 1 equal then
        1 return
    else
        N 1 minus factorial N times return
    endif.

11 factorial print.                             Use the function.
```

The postfix syntax also allows the use of applicative-style programming techniques [17,2], which allow the construction of entire programs by successively applying operators to data items, and other operators, without the usual overhead of declaring specific intermediate data structures to transfer results between routines.

Features

The language also supports access to the interpreter itself as a function, so that code can be constructed using string manipulation functions and then compiled or executed as desired. This supports both static (at compile time) and dynamic (at execution time) binding of procedures and data objects, as the user desires, which supports implementation of higher-order and abstract functions in a very natural way. Using these facilities with Thread's string manipulation capabilities, it is very easy to build very simple utilities which will manipulate fragments of programs to perform very powerful operations, such as searching sets of previously defined source code for the presence of a certain variable. The capability to define abstract functions also enables the use of object-oriented programming techniques [7], a method of maintaining extremely rigid boundaries between software components so that internal modifications and implementation details do not propagate throughout the program, but are localized for detection and correction.

Here are trivial examples of using the interpreter as a function:

`"2 2 plus print." listen_to_string.`	Call Thread to execute as string.
`"define double 2 times print." listen_to_string.` `2 double.`	Compile a definition in a string, then use it.
`Door 30.0 dg Param_value setvalue.` `Door Param_value getvalue print.`	Store and retrieve the value of the door hinge angle from the world model.

Here are examples of the use of abstract procedures which take code fragments as input parameters and thus can be implemented to function independently of any particular data type or specific application:

`List_of_drivers_licences "has_fewer_tickets" shell_sort.`
 	
`List_of_names "alphabetically_earlier" shell_sort.`
	This shell sort algorithm takes as inputs a list of arbitrary data objects and a code fragment containing an inequality relation to use during comparisons.

This same facility allows the creation of "generic" algorithms, which are only partially determined and can be instantiated to perform many different functions:

`Task_board_frame Subtree_list getvalue`	Fetch the list of primitive objects making up the task board assembly.
`"Name_string getvalue print cr"`	Create a piece of code which will display the name of a database object.
`apply_to_each_element.`	Apply the code to each element of the list.
`List_of_subtrees "length_of"` `apply_to_each_element`	Use a similar method to find the number of leaves in each subtree.
`"plus" reduce_list store Number_of_leaves.`	Add all the individual sums together into a single count.

By applying this technique to individual data objects, they can contain their own local procedures to control how they need to perform their own operations. This allows (i) the construction of tools which are very easily reconfigurable to suit new contexts, because their functions are largely determined by input data rather than built-in coding, and also allows (ii) modular segregation of *portions of algorithms* which are specific to certain data types (or situations) to be stored with those objects (or associated with that situation). An example is the use of an object-oriented database to store algorithmic information about how to compute grasp points individually for each object:

```
Door_handle1 Hinged_grasp Howto_grasp setvalue.
Panel_handle Hinged_grasp Howto_grasp setvalue.
Crank_handle1 Rotary_grasp Howto_grasp setvalue.
```

Store the appropriate hinged or rotary grasp *algorithm* (or any other code desired) with each object, as is appropriate.

```
define grasp_object
    ...
    Grasped_object Howto_grasp getvalue thread
    ...
```

To use the algorithm, retrieve it from the object model and execute it.

The language is not strongly typed, in the conventional sense, because although it performs rigid type checking, it does so entirely at runtime, which allows all variables and arguments to be generic, able to hold data of any form. This removes most of the overhead of declarations, type conversion, etc. from the user, at the cost of some computational inefficiency. The prime benefit of runtime typing, however, is that all the details of memory management and dynamic allocation are handled transparently by the language, so that objects are automatically created, copied, and destroyed whenever they should be, removing a great deal of simple, repetitive detail from the developer, at the cost of an efficiency loss. Runtime typing is also an important feature of the language because of the lack of any pointers, or reference types, which are one of the most common sources of software errors.

Another property which was designed into the language is that all data structures have direct syntactic representations, and can thus be entered at the keyboard through the interpreter at any time. This means that users never need to write special formatting routines to read or write their own data (which are built using the Thread **aggregate** type), because the language itself supports direct entry of arbitrary data structures. The language also has the capability to "decompile" either code or data back into source text which can be modified and recompiled, which will evaluate to the value of the data structure or code fragment.

An interface between the language and the host operating system is supported, which allows the user either to "shell out" to the operating system interactively and then return to the language, or to call the operating system as a subroutine to perform complex file- and program-manipulation operations. The interactive help system and many other utilities for common use were built using these facilities.

One of the prime features of the language is the extremely extensive error diagnostics, which were necessary to complement the runtime type checking, since it would otherwise be very difficult to find the location and cause of errors. This feature, combined with the decompiler and operating system interface facilities, makes it easy to obtain very rapid turnaround time during debugging.

Thread is implemented in Ada, C, Fortran and Thread itself, with its utilities and development tools implemented partially in various other interpreted languages (VMS command language, RUNOFF text processor scripts, KERMIT file transfer protocol, etc.). The capability to build this rich set of utilities comes from the fact that the other software products involved are driven by interpreted languages, which can be generated in a structured way by Thread programs and executed to perform useful work. This philosophy, of interfacing two functional modules by use of a language, rather than data structures, represents a very powerful design and implementation technique.

Using recursive calls of the interpreter itself as an input method allows the use of any available facilities, either built-in or extensions, whenever user input is being accepted or requested. There are no mode restrictions, in that any and all elements of the language are immediately available at all times, rather than being segregated into different artificial partitions which are mutually exclusive. At any time the interpreter is active, either in normal command mode or when the user has been queried to enter some information, the user can: (i) enter code at the keyboard as response to queries; (ii) access the operating system, editors, etc.; (iii) use existing utilities to generate requested inputs; (iv) override current activities; (v) access facilities at high or very low level. Typically, all geometric computations are done using the world model facilities in a convenient, interactive manner, rather than by building any specialized user-friendly interfaces.

Because of the fact that Thread is an interpreted, extensible language with a rich variety of primitive capabilities, it is very easy to use it to construct highly heterogeneous utilities which use preexisting software written in other languages. Many of the development utilities of Thread are written in other languages, and the utilities operate by calling the other languages as "subroutines", through the operating system, to perform their operations. In addition, whenever a product becomes available which is useful for robotics and which accepts input in some form of command language, it is usually quite straightforward to build a simple interface wrapper to it using Thread's abilities to call itself and the operating system as functions. A hidden-surface graphics display primitive was developed using this technique to transfer Thread graphics model structures into *Mathematica*™. This ability to so quickly and easily interface with completely separate pieces of software is probably among the most powerful features of Thread, derived in this instance from the fact that both Thread and *Mathematica*™ are fully functional languages, instead of just being specialized interactive programs to perform specific functions.

III. Robotics-Specific Features

Contributions

Because the Thread language is extensible, it has been worked on in many different ways by nearly a dozen people, many of whom contributed algorithms and suggestions about how to integrate different facilities into the language. This author developed the language concept, implemented the interpreter kernel, the development utilities, the core of the database/world model and, later, several of the high-level features such as the pseudo-English command language and the obstacle-avoiding trajectory planner. Many other individuals in the Tele-Autonomous Systems Research Group were responsible for the majority of the robotics-specific algorithms and features of the language[2].

Overview

The standard objects and operations of robotics are supported as elemental data types and primitive operations within the language. The following data types exist: homogeneous coordinate transformations [12], rotation matrices, translation vectors, N-vectors, M*N matrices, kinematics poses, sets of joint angles, solid modelling primitives (both CSG and convex polyhedra) and volume/surface/edge decompositions, graphics models, and the Thread **aggregate** data type, which can hold any collection of data objects. The basic operators available include: the standard linear algebra operators for vectors and matrices, the Singular Value Decomposition, forward/inverse kinematics based on analytical solution for the PUMA 560 manipulator (currently being generalized to recursive methods for redundant and multiple arms [14]), checking of joint stops/singularities/degeneracies, interpolation of manipulator trajectories in joint space and task space [16] (including a modification of the Taylor algorithm, developed by H. Stone, which allows pose flips to take place smoothly during joint-approximated task space motions).

Features

The language provides an active world model/database which can model both the robots and their environment using object-oriented methods for algorithmic and data abstraction, and in which commonly used functions are handled transparently by the database itself: coordinate transformations in 4 different frames associated with each object, 3-D perspective graphics (wireframe without hidden line), collision detection, using unions of both CSG [13] and of convex polyhedra, nearest-distance metrics of computational geometry [6,5], parameterized geometric relationships (hinges, etc.), general point-to-point jacobian algorithm based on Recursive Spatial Algebra [14], (in progress: consistent update using *a priori* and sensor data fusion heuristics).

Using the world model and the basic kinematics facilities, a set of motion simulators has been built to allow the user to predict the effect of singularities, jacobian manipulability measures, etc. on the outcome

[2](in alphabetical order) J. Balaram, A. Jain, K. Kreutz, B. Lau, A. Lokshin, B. Mueller, R. Sidney, H. Stone

of a motion:

```
Bolt2 Abslocal_trnsf getvalue

10.0 cm up

RUN Task_interpolation Move_Absolute.
```

Get the coordinates of the bolt from the world model. Offset them by an appropriate amount to hover above the bolt.
Use Right-arm elbow-Up Negative-wrist kinematics pose and cartesian interpolation mode.
Simulate a motion to that point.

A basic graphics capability is available for animation, using 2-D plots and 3-D wireframe models without hidden line removal, but with a very user-friendly interface for controlling viewing conditions of the world model. (More sophisticated hidden-line graphics, both animated and hardcopy, are available through interfaces to other products, such as *Mathematica*™ and the IRIS 4D-70 GT graphics engine.)

For purposes of high level planning, simulators which predict the kinematical behavior of the robots for several specific actions have been developed, including grasping of an object, swinging a door, or contacting a surface while performing guarded motion.

```
Task_board 10.0 cm Z_direction move_contact.
```

Move 10 cm in the Z direction, expecting to contact the task board.

```
Crank_bar 30.0 dg 3.0 dg swing_object.
```

Rotate the crank bar by 30 degrees at a rate of 3 degrees per second.

One of the high-level utilities built into the language is a simple heuristic search path planner, which will find obstacle-free paths which also satisfy all other kinematic restrictions. The planner usually finds a path within one minute for an uncluttered environment which requires only one or two straight line (joint or cartesian) segments to negotiate. The path planner functions using a heuristic combinatoric search through choice-sets of positions, kinematics poses, and joint and cartesian interpolation modes [16], calling on whatever objects are in its copy of the world model to perform kinematic constraint checking. This very naturally allows the path planner to be used for a variety of purposes in addition to free-motion planning, such as choosing a grasp point on a handle which will permit the successful opening of a door, automatically taking into account the additional motion of the door for the simple reason that the world model registers it as being attached to the hand.

```
Crank_handle1 Abslocal_trnsf getvalue
15.0 cm up
Left_elbow_up_poses
build_choice_set
plan_a_path
move_the_arm.
```

Fetch the coordinates of the crank handle, offset them to the desired position near the handle, choose which kinematics poses of the arm to use, build the combinatoric set for the path planner to search, then use the planner's result to move the arm.

The Run Time Controller's highest level of functionality is a process planner and pseudo-English command parser which provides a front end to all existing RTC autonomous capability. It is, of course, available at all times to perform convenient services for the researcher experimenting in the laboratory. One common use is to employ the command language to avoid the details of controlling each individual motion of the arm:

```
grasp the Door_handle with the right arm
    at a location 2.5 cm below its top
    with an orientation of 90.0 dg left_tilt
    using compiliance
    please.
```

Self-explanatory.

One of the most important class of primitive functions in the language is the set of facilities which permits communication with the external subsystems devoted to manipulation and sensing. These primitives are so important because they permit the user to actually perform physical actions, either via software or manually through the interactive capabilities of Thread. These interfaces were implemented in the JPL Telerobot using a customized Network Interface Package (NIP) which supports definition of transaction protocols in real time by the various subsystems. This is obviously a very specialized communications interface, but it has been very simply and modularly integrated into the Thread interpreter, using facilities which permit the user to build communication packets on a byte-by-byte basis, which may then be transmitted and received using either the NIP or whatever other method is desired. Alternative communications protocols have in fact been implemented in Thread for various purposes, the simplest being a KERMIT-based protocol which allowed Thread software to interface through a modem to a personal computer, to create a Thread primitive called `prolog` which accepts Prolog language source code as a text string and remotely compiles and executes it on the personal computer, returning the result as another text string.

IV. Conclusion

Over the last two years, Thread has had roughly a dozen users who have produced a total of over thirty thousand lines of Thread code, for many different applications, and Thread has proven to be quite useful both for off-line algorithm development and especially for experimental work with actual manipulators.

Unfortunately, another lesson learned during the development of the Run Time Controller is that the magnitude and severity of the software development task for large systems such as the Telerobot is still not taken seriously by some of the robotics community [15].

The original decision to use Thread as an interactive development operational environment led us to a much more cooperative man-machine viewpoint on the Telerobot than is typical for either traditional robotics or teleoperations, and this has had a strong impact on our understanding of both autonomous and teleoperated systems. One observation made is that the benefits in capabiliity which accrue from sharing functions between the human and the robot are very signficant: a dual-arm single-joystick shared control facility was demonstrated to be able to allow the human to guide two arms simultaneously under master-slave compliant control, to perform motions which could not realistically be planned by an autonomous system, with a degree of delicacy (forces applied to the carried object) which would be impossible to a human two-handed teleoperation system.

The increase in capability which occurs through sharing becomes even more drastic at higher levels than that of servo control, however, and also has the added advantage for space applications [1] that it is robust under time delay, which force-reflecting teleoperation is not. The interactive nature of the Thread development environment led to many instances in the laboratory when it was relatively easy for a member of our software development team to directly use the individual subsystem components (trajectory planner, world model, sensing/manipulation subsystem interfaces, etc.) to directly control the robot in an interactive manner to perform tasks which were beyond the ability of both the planning and reasoning capability of the existing autonomous robot, and beyond the dexterity of the teleoperation system.

Also, the interactive, multilayered architectural structure which Thread encourages is not at all restricted to non-real-time planning software, as a prototype extension of Thread to an interactive interpreter which allows complete control of servo-loop behavior at the lowest level has been built and successfully tested.

This work indicates that the proper direction for practical telerobotics [11] is to pursue a much more cooperative man-machine style of interface than either traditional supervised robotics or pure teleoperation.

References

[1] J. S. Albus, H. G. McCain, R. Lumia, "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", *Technical Report, National Bureau of Standards, Robot Systems Division*, December 1986.

[2] J. Backus, "Function Level Programs as Mathematical Objects", *ACM Transactions on Programming Languages*, October 1981.

[3] F. P. Brooks Jr., "The Mythical Man-Month (Essays on Software Engineering)", *Addison-Wesley*, New York, 1982.

[4] M. Ewing, "The Caltech FORTH Manual", *Owens Valley Radio Observatory Technical Publication*, 1980.

[5] B. Faverjon, P. Tournassoud, "A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom", *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pages 1152-1159.

[6] E. G. Gilbert, D. W. Johnson, "Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, March 1985.

[7] A. Goldberg, "Smalltalk-80: The Language and Its Implementation", *Addison-Wesley Series In Computer Science*, 1983.

[8] A. Goldberg, "Smalltalk-80: The Interactive Programming Environment", *Addison-Wesley Series In Computer Science*, 1983.

[9] W. A. Gruver, B. I. Soroka, J. J. Craig, T. L. Turner, "Industrial Robot Programming Languages: A Comparative Evaluation", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-14, No. 4, July/August 1984.

[10] V. Hayward, R. Paul, "Robot Manipulator Control Under UNIX RCCL: A Robot Control 'C' Library", *International Journal of Robotics Research*, Vol. 5, No. 4., pages 94-111, Winter 1987.

[11] M. D. Montemerlo, "The Space Perspective: Man-Machine Redundancy In Remote Manipulator Systems", *Keynote speech, NATO Advanced Research Workshop on "Robots with Redundancy: Design, Sensing and Control"*, June 27-July 1, 1988, Salo, Lago di Garda, Italy.

[12] R. Paul, "Robot Manipulators: Mathematics, Programming, and Control", *MIT Press*, 1981.

[13] A. A. G. Requicha, R. B. Tilov, "Constructive Solid Geometry", *Technical Memorandum 25, Production Automation Project*, University of Rochester, 1977.

[14] G. Rodriguez, K. Kreutz, A. Jain, "A Spatial Operator Algebra for Manipulator Modeling and Control", *Proceedings of the 1989 NASA Conference on Space Telerobotics*, JPL Publication 89-7 (this proceedings).

[15] H. W. Stone, J. Balaram, J. Beahan, "Experiences with the JPL Telerobot Testbed - Issues and Insights", *Proceedings of the 1989 NASA Conference on Space Telerobotics*, JPL Publication 89-7 (this proceedings).

[16] R. Taylor, "Planning and Execution of Straight-line Manipulator Trajectories", *IBM Journal of Research and Development*, 23, 1979.

[17] D. Turner, "The Semantic Elegance of Applicative Languages", *ACM Transactions on Programming Languages*, October 1981.

[18] M. I. Vuskovic, "R-SHELL: A UNIX-Based Development Environment for Robotics", *San Diego State University Technical Report SDSU-CS-RL-03-87*, October 1987.

[19] P. H. Winston, B. K. P. Horn, "LISP", *Addison-Wesley*, Philippines, 1981.

MULTI-ARM CONTROL

Stability Analysis of Multiple–Robot Control Systems

John T. Wen
ECSE Department
Rensselaer Polytechnic Institute
Troy , NY 12181

Kenneth Kreutz
Jet Propulsion Laboratory /
California Institute of Technology
4800 Oak Grove Dr., M/S 198–330
Pasadena, CA 91109

Abstract

In a space telerobotic service scenario, cooperative motion and force control of multiple robot arms are of fundamental importance. In this paper, we propose three paradigms to study this problem. They are distinguished by the set of variables used for control design; the three possibilities are: joint torques, arm tip force vectors and the acceleration of a set of generalized coordinates. Control issues related to each case are discussed. The latter two choices require complete model information, which presents practical modeling, computational and robustness problems. We therefore focus on the joint torque control case to develop relatively model–independent motion and internal force control laws. The rigid body assumption allows the motion and force control problems to be independently addressed. By using an energy motivated Lyapunov function, we show that a simple proportional derivative plus gravity compensation type of motion control law is always stabilizing. The asymptotic convergence of the tracking error to zero requires the use of a generalized coordinate with the contact constraints taken into account. If a non–generalized coordinate is used, only convergence to a steady state manifold can be concluded. For the force control, both feedforward and feedback schemes are analyzed. The feedback control, if proper care has been taken, exhibits better robustness and transient performance.

1. Introduction

Spacecraft servicing by using autonomous telerobots has been under serious consideration for future deployment, such as the flight telerobotic servicer concept currently under study in NASA. In a typical telerobotic service scenario, a number of challenging control problems arise, including the control of open kinematic chains (arms moving into ready positions for servicing) and closed kinematic chains (arms handling a satellite, manipulating parts etc.), and attitude control (attitude of the platform that supports the arms, and attitude of the satellite). In this paper, we will address the issues related to the cooperative control of multiple rigid robot arms holding a commonly held object that is possibly in contact with a rigid surface.

A multiple–arm system can be viewed in different ways depending on the variables regarded as the control input in the controller design. Three levels of control paradigms can be constructed. On the first level, the joint torques are viewed as the control input. We call this perspective the full dynamics approach. On the second level, the tip forces are regarded as the control input and the joint torques are selected in a feedforward manner (which still requires real time joint angle measurements but has no error correction function) to realize the prescribed tip forces. We call this perspective the arms–as–actuator approach. On the third level, an unconstrained generalized acceleration (there are an infinite number of generalized coordinate representations for the constrained dynamics) is set equal to a pseudo–control input and the joint torques again generate the prescribed control action via a feedforward compensation. We call this perspective the feedback linearization approach.

By the nature of their structures, the last two approaches require the full dynamical model information to implement the feedforward compensation. However, the control law design is much simplified as the nonlinear dynamics of the arms are compensated. Since computational and robustness issues related to the multiple–arm control problem remain to be fully explored, we will focus on the full dynamics approach in this paper.

Due to the rigidity assumption on the held object, the grasp and the arms themselves, it is possible to decompose the tip contact force (of all arms collectively) into two orthogonal components, one that effects

motion of the held object and the other that produces a desired internal squeeze force. As a consequence, the motion and force control problems become decoupled (in one direction) in the following sense:

Force control does not affect object motion; object motion does affect the internal force (due to the inertial, d'Alembert force).

This motivates the following control design philosophy:

Design a stable motion control law without the consideration of force control. Then design a stable force control law by treating the inertial force as a perturbation.

Based on this philosophy, motion control laws for set point operation, with and without transient shaping, are developed by using a class of Lyapunov functions motivated by the total energy of the system. This class of control laws has an appealing simple structure of proportional and derivative feedback with gravity compensation. In the set point control of the internal force, both feedforward or feedback (if force/torque sensor information is available) strategies are analyzed. The feedback scheme has better robustness properties than the feedforward one, and, because of the motion/force decoupling property mentioned before, it can achieve tight transient control with high feedback gains.

For motion control, several choices of feedback variables are possible, with different implications in terms of performance, stability, relation to the control objective, amount of model information needed in implementation, etc. Here, we consider the joint variable, the tip variable and an unconstrained generalized coordinate. In all three cases, the proportional–derivative–gravity control law drives the system to a steady state configuration. However, only in the case of generalized coordinate does the steady state configuration correspond to the desired one. In the first two cases, the configuration lies in a manifold on which the tip forces produced by the arm position errors balance with one another. We call this manifold the jam manifold. Some preliminary results of its properties are discussed.

2. Model for Multiple–Arm Systems

All of the stability results discussed in this paper are based on the assumption that the arms and the held object are rigid and the grasp between the arms and the object is also rigid. Other models of multiple–arm systems sometimes insert a spring in the last link of each arm to simulate the effect of force/torque sensors. We feel that because the internal spring in the force/torque sensor is sufficiently rigid (implying small displacement) and the anticipated force transient is sufficiently benign (due to our force controller), our infinite rigidity assumption is a reasonable approximation. With the additional assumption that the object does not deform, we can decompose the tip force vector into two orthogonal components: one that contributes to motion of the system and one that builds up internal force. The component that effects motion is said to be in the "move" subspace and the component that builds up internal force is said to be in the "squeeze" subspace. This decomposition is appealing for several reasons. It agrees with human experience that squeeze forces can be applied without effecting any apparent motion. The analysis is simpler since it is free from the added complication of a spring in every arm, the effect of which does not appear to be very significant physically (if the spring is due to the force/torque sensor only and the internal force is controlled). There is also the possible application to task separation in combined autonomous/teleoperated types of operation (e.g. autonomous positioning and teleoperator force control). An important consequence of the rigidity assumption is that the motion and force control problems can be decoupled. The squeeze force control does not affect the motion of the held object, but the motion can affect the squeeze force. This motivates the following approach to control design: Design motion control first independent of the force control, then design force control by treating motion induced squeeze force (projection of the d'Alembert force in the squeeze subspace) as an external perturbation (which is unaffected by the squeeze control effort). The rigidity assumption also prevents direct proportional force feedback, for the algebraic loop results in an ill–posed dynamical system, destabilized by arbitrarily small delay in the force feedback channel; a filter with memory must be used instead. This issue will be addressed later in the section on force control.

With the assumptions stated above, the equation of motion are [1]:

$$M\ddot{q} = \tau - C\dot{q} - k - J^T f \tag{2.1a}$$

$$f_c = A^T f \tag{2.1b}$$

$$M_c \alpha_c = f_c + b_c + k_c \tag{2.1c}$$

$$\alpha = A\alpha_c + a = \dot{J}\dot{q} + J\ddot{q} \tag{2.1d}$$

$$v = Av_c = J\dot{q}$$

The symbols are defined as follows (an arm-related vector is composed of stacked single-arm vectors; an arm-related matrix is composed of block diagonalized single-arm matrices): q = joint variable, M = inertia matrix, C = arm coriolis and centrifugal force, k = arm gravity load, τ = joint torque, f = tip force, J = arm Jacobian, A = object center of mass (CM) to arm tip Jacobian, f_c = force at object CM, b_c = object coriolis and centrifugal force, k_c = object gravity load, α = tip acceleration, α_c = object CM acceleration, v = tip velocity, v_c = object CM velocity, a = bias arm tip acceleration.

These equations can be combined to solve for the contact force, f, as

$$f = (AM_c^{-1}A^T + JM^{-1}J^T)^{-1}(\dot{J}\dot{q} - a - AM_c^{-1}(b_c + k_c) + JM^{-1}(\tau - C\dot{q} - k)) \quad . \tag{2.2}$$

f is uniquely solvable if and only if $[J \quad A]$ has full rank. It was shown in [Cor.3.3 in 2] that if the manipulator system is kinematically parameterized by the Denavit-Hartenberg parameters and base positions of the arms, then $[J \quad A]$ having full rank at every kinematically feasible configuration is a generic property. Hence, we will assume that f is uniquely solvable. For the ease of presentation, we restrict our attention to the non-redundant arms case, though the redundant arms case can also be considered in the same framework.

The matrix A^T in (2.1b), which transforms the tip force to the force at object CM, is "fat", and therefore possesses a nontrivial null space. Since $f_c = A^T f$, f in this null space means that it does not contribute to the motion of the held object but only to the buildup of internal forces. Hence, we define the squeeze subspace to be $\mathbf{X}_s = \mathrm{Ker}(A^T)$ (the kernel or the null space of A^T). The orthogonal complement of $\mathbf{X}_s$ is defined as the move subspace, which is $\mathbf{X}_m = \mathrm{Im}(A)$ (the image or the range space of A) since $\mathbf{R}^{m \cdot n} = \mathbf{X}_m \overset{\perp}{\oplus} \mathbf{X}_s$. For a given tip force, f, there exists a unique orthogonal decomposition

$$f = f_m + f_s$$

where $f_m \in \mathbf{X}_m$ and $f_s \in \mathbf{X}_s$. Only f_m contributes to the motion of the held object. A^T can be written as

$$A^T = [A_1^T, \ldots, A_m^T]$$

where A_i^T transforms the tip force of arm i to the force at the object CM, and it is given by

$$A_i^T = \begin{bmatrix} I & \tilde{r}_{iC} \\ 0 & I \end{bmatrix}$$

where r_{iC} is the vector from the tip of arm i to the object CM and $\tilde{}$ denotes cross product in a coordinate representation. Clearly, A_i^T is non-singular, and, hence, A^T is of full row rank. This implies that $\dim(\mathbf{X}_s) = (m-1) \cdot n$ and $\dim(\mathbf{X}_m) = n$.

In (2.1 b-d), the tip forces of the arms can be regarded as actuator outputs. This leads to the arms-as-actuator approach. Eq. (2.2) can then be considered as a nonlinear compensator which computes the joint torques needed for the desired actuation signal. This viewpoint has also been adopted in [3].

For a selected set of unconstrained generalized coordinates, the multiple-arm dynamical equation can be partitioned in a different way so that the generalized acceleration is equal to a desired value. A nonlinear

feedforward filter then computes the joint torques required for the desired behavior. Denote the generalized coordinate and its kinematic relation to the joint angles by

$$\beta = h(q) \quad . \tag{2.3}$$

Then by differentiating twice with respect to t and denoting $J_\beta(q) \triangleq \nabla_q h(q)$, we have

$$\ddot{\beta} = u \tag{2.4a}$$

$$u = \dot{J}_\beta \dot{q} + J_\beta M^{-1}(\tau - C\dot{q} - k - J^T f) \quad , \tag{2.4b}$$

where f is given by τ, q and $\dot{q}$ as in (2.2) . We call the perspective of regarding u as the effective control input the feedback linearization approach. Note that (2.4) is valid irrespective of the redundancy of the arms.

3. Control Issues Related to the Feedback Linearization and the Arms–As–Actuator Approaches

In the arms–as–actuator paradigm, the dynamics involving the held object seen at the center of mass are composed of two parts: a force balance equation (Newton's equation) and a torque balance equation (Euler's equation). The force equation is linear and can be controlled easily. The torque equation involves control on the rotation group, $SO(3)$. Various possible control laws for the latter problem have been analyzed in [4,wkattcdc]. In particular, a control law involving the feedback of the unit quaternion of the attitude error can be used for globally asymptotically stable closed loop operation.

In the feedback linearization paradigm, the control law can be easily constructed since the feedback linearized system is in double integrator form. However, J_β in the control law introduces additional singularities which are a mathematical constraint rather than a physical limitation.

The non–uniqueness of f, and therefore τ, for controlling the object motion can be used to control the closed chain internal forces. This can be posed as an optimization problem, giving rise to the load–balancing problem as discussed in [6].

Full dynamical model information is needed in the nonlinear feedforward compensation for both the feedback linearization and arms–as–actuator schemes. The computational and robustness issues due to the complex nonlinear, model dependent compensation need to be addressed for a successful implementation. At the present, work in this direction is lacking in the context of multiple–arm systems. For this reason, the rest of the paper will focus on the full dynamics approach and develop relatively model–independent control laws directly for the joint torques.

4. Control Issues Related to the Full Dynamics Approach

In this section, the full dynamical model is analyzed to develop motion and force control strategies. A consequence of the move/squeeze decomposition is that any term in τ of the form $J^T F_s$, with F_s in the squeeze subspace, does not affect motion of the system. However, the motion of the system does affect the actual squeeze force, due to the squeeze component of the d'Alembert (inertial) forces. This motivates the following decomposition of the control torque:

$$\tau = \tau_m + \tau_s + \tau_g \quad , \tag{4.1}$$

where τ_m is responsible for the motion control, τ_s is responsible for the squeeze force control and τ_g compensates for the gravity load due to the arms and the held object. In Section 4.1, various possible motion feedback control laws, based on the variables used for feedback, are discussed. In Section 4.2, different force control laws are discussed.

For motion control, we propose a class of relatively model independent control laws that have a simple Proportional Derivative (PD) plus gravity compensation type of structure. For the internal force control,

asymptotically stable, model independent, set point controllers based on feedforward or feedback implementation are constructed. The feedback controller is shown to possess superior robustness property and transient performance.

4.1 Motion Control

It has been shown [7] that PD feedback plus gravity compensation is a globally asymptotically stable set point control law for a single arm with an unconstrained tip. The feedback variable can be either the joint variable (angle or displacement) or the tip variable (tip position and orientation, the latter being suitably parameterized). For tip variable feedback, Jacobian non–singularity for all time is assumed. The structure of this class of control law is very appealing since it is relatively model independent (only arm gravity compensation is needed) and has an energy dissipation interpretation. Therefore, it is reasonable to investigate its generalization to the multiple–arm case.

For a multiple–arm system, there are three possible types of feedback variables: 1. joint position (of all arms), 2. tip position (of all arms), 3. a generalized coordinate. The first two over–specify the configuration of the system (due to the kinematic constraint imposed by the rigid grasp of a common object) and hence are not generalized coordinates. The ramification of using them for feedback will be discussed later. Possible choices for generalized coordinates include position and orientation of the mass center of the held object, a subset of the tip position, joint position and/or tip forces. For tip position feedback and generalized coordinate feedback involving orientation, a parameterization of orientation needs to be chosen. We will assume that a minimal representation is used, though other related works [4,5] have indicated that the unit quaternion (Euler parameters) may be a better choice since there is no problem with the singularity of representation.

We will address point–to–point control only. Generalization to the general tracking problem is currently under investigation. If the transient performance is not expressly considered, then, as shown in [8,9], a straightforward generalization of the single arm energy Lyapunov function approach shows that a steady state is reached when PD plus gravity type of control law is used for all three types of feedback variables. In the cases of joint level and tip level feedback, the steady state error converges to a manifold, even if the set point represents a kinematically feasible configuration. Only in the generalized coordinate feedback case is the steady state error zero.

The fixed–set–point control laws in [8,9] are useful in demonstrating the application of a general class of Lyapunov functions and pointing out some interesting issues unique to the multiple–arm control problem (jam manifold, squeeze force control, etc.). However, the fixed set point control paradigm is fundamentally flawed because the closed loop trajectory transient is not controlled. For initial condition far away from the desired set point, the transient is typically so wild, these control laws are virtually unusable. The problem is most severe in tip and generalized coordinate feedback, where arms may cross Jacobian singularities, flip poses (due to multiple solutions to the inverse kinematics problem), collide with themselves, violate joint stops, etc. This motivated us to extend our framework to include trajectory shaping in the set point operation.

A possible method to shape transient performance is to replace the position error and the velocity in the fixed–set–point control laws by the difference between the actual trajectory and a desired trajectory to the goal set point which is chosen to have good transient behavior. Since the desired trajectory converges to a set point, we shall call it the moving set point. Intuitively, we expect better transient response with the moving set point controller since the applied torque increases gradually rather than abruptly. In this section, we will show that provided the desired velocity and acceleration satisfy some mild conditions, the moving set point controller results in the same closed loop stability property as in the fixed set point case.

Assume the desired trajectory has the following properties:

$$\Big\{q_{des}(t)\rightarrow \text{ constant },\ \dot{q}_{des}(t)\rightarrow 0,\ \ddot{q}_{des}(t)\rightarrow 0,$$
$$\dot{q}_{des}\in L_2,\ \ddot{q}_{des}\in L_2\cap L_\infty,\ \text{ both } \ddot{q}_{des}, \dddot{q}_{\text{des}} \text{ are continuous }.\Big\}$$

These are mild restrictions on the desired trajectories. If the desired trajectory possesses these properties, then we say the desired trajectory belongs to the class $\mathcal{S}$. We modify the Lyapunov function used for fixed-set-point control by replacing velocities by velocity tracking errors:

$$V=\frac{1}{2}\Delta v_c^T M_c\Delta v_c+\frac{1}{2}\Delta\dot{q}^T M\Delta\dot{q}+U^* \tag{4.2}$$

where

$$\Delta v_c=v_c-v_{c_{des}},\quad v_{c_{des}}=A^+v_{des},\quad v_{des}=J(q)\dot{q}_{des},\quad \Delta\dot{q}=\dot{q}-\dot{q}_{des}\quad ,$$

and U^* is given below

Type of Feedback	$U^*(q)$
Joint Level	$\frac{1}{2}\Delta q^T K_p\Delta q$
Tip Level	$\frac{1}{2}\Delta x^T K_p\Delta x$
Generalized Coordinate	$\frac{1}{2}\Delta\beta^T K_p\Delta\beta$

Table 1. Quadratic Potential Energy Candidates

The generalized velocity, $\dot{\beta}$ is related to the tip velocity via $v=B\dot{\beta}$. The superscript $^+$ denotes Moore-Penrose generalized inverse. Suppose the following proportional-derivative-gravity control law is used for the motion control (cf. (4.1)):

$$\tau_m+\tau_g=-\tau_p-\tau_v+k(q)-J^TF_c\quad , \tag{4.3}$$

where k is the arm gravity load, and F_c is gravity compensation for the held object, chosen to satisfy $A^TF_c=k_c$. The proportional and velocity feedback terms are given in the table below, depending on the variables used for feedback:

Type of Feedback	τ_p	τ_v
Joint Level	$K_p\Delta q(t)$	$K_v\Delta\dot{q}(t)$
Tip Level	$J^TK_p\Delta x(t)$	$J^TK_v\Delta v(t)+D\Delta\dot{q}(t)\quad D>0$
Generalized Coordinate	J^TF_P	$J^TF_D+D\Delta\dot{q}(t)\quad D>0$
	$B^TF_D=K_p\Delta\beta(t)$	
	$B^TF_D=K_p\Delta\dot{\beta}(t)$	

Table. 2 PD Feedback in Moving Set Point Motion Control Laws

Then the derivative of V along the solution can be bounded by

$$\dot{V}=-\lambda\|\Delta\dot{q}\|^2+\eta_1(t)\|\Delta\dot{q}\|^2+\eta_2(t)\|\Delta\dot{q}\| \tag{4.4}$$

where $\eta_1(t)\rightarrow 0$ as $t\rightarrow\infty$ and $\eta_2\in L_2[0,\infty)$ due to the assumed properties of the desired trajectory. By integrating both sides from t_o to t, for t_o sufficiently large, there exists $\lambda_1>0$ such that

$$\begin{aligned}\lambda_1\|\Delta\dot{q}\|^2_{L_2([t_o,t])}&\le V(t_o)-V(t)+\int_{t_o}^{t}\eta_2(\tau)\|\Delta\dot{q}(\tau)\|\,d\tau\\&\le V(t_o)+\|\eta_2\|_{L_2([t_o,t])}\|\Delta\dot{q}\|_{L_2([t_o,t])}\quad .\end{aligned} \tag{4.5}$$

Now, by completing the squares involving $\|\Delta\dot{q}\|_{L_2([t_o,t])}$, it follows that $\Delta\dot{q} \in L_2([t_o,\infty))$. From (4.5) , $V(t)$ is uniformly bounded for all t, which implies Δq, $\Delta\dot{q}$, $\Delta\ddot{q}$ and $\Delta\dddot{q}$ are uniformly bounded. By Barbalat's Lemma [10], $\Delta\dot{q}\rightarrow 0$ as $t\rightarrow\infty$, which, by Lemma 1 of [11], implies $\Delta\ddot{q}\rightarrow 0$, also. Now, from the arm dynamical equation, $\tau_p + J^T F_c + J^T f \rightarrow 0$ which yields the same convergence result as in the fixed set point case.

The stability properties of the moving set point controllers can now be summarized below:

Result 4.1. If the desired trajectory belongs to class S, then the multiple–arm system with control laws (4.3) and Table 2 has the following stability property:

Type of Feedback	Type of Stability
Joint Level	$\dot{q}\rightarrow 0$, Δq converges to the manifold $K_p\Delta q + J^T(F_c + f) = 0$ (4.6)
Tip Level	$v\rightarrow 0$, Δx converges to the manifold $J^T(K_p\Delta x + F_c + f) = 0$ (4.7)
Generalized Coordinate	β, $\dot{\beta} \rightarrow 0$ (global asymptotic stability).

Table. 3 Stability Properties of Fixed Set Point Motion Controllers

Furthermore, if the initial tracking error is zero, the maximum trajectory tracking error is inversely proportional to the size of the PD gains.

■

If the desired trajectory starts with the same initial condition as the actual trajectory and has the desired transient behavior, e.g., small overshoot, no excessive acceleration or jerk, avoiding Jacobian singularities, joint stops, obstacles and the arms themselves, Result 4.1 shows that high enough feedback gains ensure that the actual trajectory will have similar properties, also. The maximum tracking error can be shown proportional to the L_2–norm of η_2 in $\dot{V}$ which is composed of the difference between the moving set point and its steady state, the desired velocity and acceleration. A trajectory planning problem can be posed to find a desired trajectory that satisfies the required transient response and minimizes the L_2–norm of η_2.

Control laws that incorporate the full model information can also be constructed within this approach (by using, for example, results in [12]). We can qualitatively state the advantage of this added complexity in the control algorithm. In the tracking control problem, even if the initial tracking error is zero, the PD control law will always incur a nonzero trajectory tracking error. This error can be made small if gains are allowed to be large; however, it may not always be practical, given the limited actuator size and the noise problem. With the model–dependent control laws, the tracking error will remain zero (at least theoretically; noise will cause small deviation from the desired trajectory). The same is true in the internal force control (see next section). If the full model information is assumed, precise force control at every moment in time is possible; while the model–independent control law reduces finite force error with high gains. The type of control law to use for a given application depends on the trade–off between the available a priori model information and the performance requirement, subject to actuator and sampling constraints.

4.2 Force Control

In this section, we consider the problem of choosing τ_s in (4.1) to asymptotically drive the squeeze force to a desired set point. To ensure that arm motion is not affected by the squeeze force control, we choose

$$\tau_s = J^T F_s \tag{4.8}$$

where F_s is restricted to lie in the squeeze subspace. The effective control variable for force control is now F_s.

Without using the full model information, the squeeze force can only be controlled asymptotically. (If model information is available, an additional optimal load distribution problem can be posed; see [6].) We will show that either feedforward or feedback control structure may be used to drive the squeeze force to its set point asymptotically, depending on whether force sensors are available. For the full composite force vector, m force sensors need to be used. The feedback strategy, if properly applied, has better performance and robustness. If the gravity load in the squeeze subspace is not fully compensated, feedback control can still be used but the feedforward control will incur a squeeze force error equal to the squeeze component of the gravity error.

By projecting the composite tip force vector, given by (2.2), in the squeeze subspace, and applying (4.8) , we have

$$f_s = F_s + \eta \tag{4.9}$$

where η represents the projection of the inertial force in the squeeze subspace. Recall that an important property of η due to the move/squeeze decomposition is that it is not affected by F_s. Hence, it can be treated as an external disturbance. We assume that one of the PD type of control strategies in Section 4.1 has been used for motion control. (In fact, any stable motion control law can be used without affecting the subsequent argument.) Then $\eta(t) \to 0$ as $t \to \infty$. We are interested in studying the following aspects of the force control problem:

1. Stability. (Does $f_s(t) \to f_{s_{des}}$ as $t \to \infty$ in (4.9) ?)
2. Transient performance. (What is the maximum force error, i.e., $\max_{t \geq 0} f_s(t) - f_{s_{des}}$?)
3. Convergence rate. (How fast does $f_s \to f_{s_{des}}$?)
4. Noise reduction. (If $\eta(t) \to \eta_\infty \neq 0$, representing a persistent noise, what is the steady state force error?)

To attain asymptotic stability, a feedforward control will clearly suffice

$$F_s = f_{s_{des}} \quad . \tag{4.10}$$

However, the transient performance and convergence rate are determined entirely by η (which are in turn determined by the quality of the motion control law). There is also no noise reduction in this scheme.

If the arm tip forces are measured, then clearly a feedback strategy is preferable, due to the hope for added insensitivity to noise and improved transient performance. However, the infinite rigidity assumption stated in section 2.1 necessitates extra care in the control design. We will show that the lack of dynamics in (4.9) means that infinite bandwidth feedback from f_s to F_s would violate the strict causality of the loop. This has some unintended consequences. For example, the control law

$$F_s = f_{s_{des}} + \beta(f_s - f_{s_{des}}) \tag{4.11}$$

implies $f_s \to f_{s_{des}}$ for $\beta \neq 1$. Furthermore, transient performance, convergence rate and steady state error due to noise can all be much improved over the feedforward case, if β is large. However, an arbitrarily small time delay in the feedback channel (which is always present in a physical implementation) leads to instability if $|\beta| > 1$. If $|\beta| < 1$, then the response of the resulting linear discrete time system consists of two terms: the homogeneous solution and the particular solution due to η. For fast convergence of the homogeneous solution to zero, β needs to be close to zero, but then the response is similar to that of the feedforward control and the desirable properties due to the force feedback is lost.

Recognizing that the problem is caused by the algebraic loop due to the proportional force feedback, we suggest pre-processing the measured force by a strictly causal filter (if the filter is linear, then strictly proper). The feedback control law then takes on the following form:

$$F_s = f_{s_{des}} + C(f_s - f_{s_{des}}) \quad , \tag{4.12}$$

where C is a strictly proper linear filter such that $(I - C)$ has zeros only in the open left half plane. Clearly, $f_s(t) \to f_{s_{des}}$ as $t \to \infty$. To see the transient behavior, we write

$$\Delta f_s \triangleq = f_s - f_{s_{des}} = \mathcal{L} * \eta \tag{4.13}$$

where $*$ denotes the convolution operator and $\mathcal{L}$ the convolution kernel associated with $(I - C)^{-1}$. Since $\mathcal{L}$ can contain arbitrarily fast dynamics (if the desired dynamics of $\mathcal{L}$ is $\frac{a(s)}{b(s)}$ in the Laplace domain, then $C(s) = \frac{a(s)-b(s)}{a(s)}$ is the Laplace transform of the corresponding filter C), the L_1 norm of $\mathcal{L}$ can be made arbitrarily small. By the following error estimate [Appendix C,13],

$$\|\Delta f_s\|_{L_\infty} = \|\mathcal{L}\|_{L_1} \|\eta\|_{L_\infty}$$

it follows that the transient performance and convergence rate can both be improved. To see the effect of a persistent noise, we apply the initial value theorem:

$$\lim_{s \to 0} s \Delta f_s(s) = \lim_{t \to \infty} \Delta f_s(t) \quad . \tag{4.14}$$

We conclude that if $C(s)$ has a pole at the origin, then there is no steady state error even if η does not converge to zero. Hence, all the control objectives are satisfied with the control law (4.12) , provided that $(I - C)^{-1}$ is a stable filter and C has a pole at the origin. If the spectrum of η is known (say, for repeated tasks), C can be chosen to selectively notch out the dominant dynamics in η. What about robustness with respect to small time delays? To address this problem, we use a first order approximation of $f_s(t - \Delta t)$, for Δt small, i.e.,

$$f_s(t - \Delta t) \approx f_s(t) - \Delta t \dot{f}_s(t) \quad .$$

The closed loop system in the Laplace domain now becomes

$$\Delta f_s(s) = (I - C(s) - \Delta t C(s) s)^{-1} \eta(s) \quad . \tag{4.15}$$

Since $C(s)$ is strictly proper and $(I - C)^{-1}$ is stable, for sufficiently small Δt, the perturbed system (4.15) remains stable. In the case of direct proportional feedback, C is not strictly proper and indeed the corresponding closed loop system becomes unstable for arbitrarily small Δt.

A particularly simple choice of C is just an integrator, i.e., in the Laplace domain,

$$C = -\frac{\beta}{s} \quad . \tag{4.16}$$

This control law has all the desirable features discussed above. If the integral feedback gain β is chosen sufficiently large, and $\dot{\eta}(t)$ is uniformly bounded in t, then by explicitly solving the closed loop dynamical equation, it can be shown that the transient effect of η on $f_s - f_{s_{des}}$ can be made arbitrarily small.

The discussion in this section can be summarized in the result below:

Result 4.2. For the multiple–arm control system under consideration, if the arm configuration converges to a steady state (i.e., velocity converges to zero), then either the feedforward controller (4.11) or the feedback controller (4.12) , with C a strictly proper linear filter and $(I - C)$ containing zeros only in the open left half plane, drives $f_s \to f_{s_{des}}$.

If in (4.12) , C has a pole at the origin, then replacing τ_g in (4.1) by its projection in the move subspace does not affect the asymptotic convergence of $f_s - f_{s_{des}}$ to zero, and, in general, $f_s \to f_{s_{des}}$ even if $\eta \to \eta_\infty \neq 0$.

If C is chosen to be an integrator as in (4.16) and η in (4.15) has a uniformly bounded time then $f_s(t) - f_{s_{des}}$ tends to zero uniformly for t in bounded intervals as $\beta \to \infty$.

■

5. Conclusion

This paper has considered several possible control structures for multiple–arm systems by regarding either joint torques, tip force, or a generalized acceleration as the control input. We have mainly focused on the first case since a class of relatively model–independent control laws can be generated for both motion and internal force control. The recently developed move/squeeze orthogonal subspace decomposition coupled with the energy Lyapunov function formulation provides the basic analytical framework. Future research includes the tuning of PD gains to improve tracking performance and generalizations to the multiple degrees–of–freedom contact case.

Acknowledgment

The research described in this paper was carried out in part at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Bibliography

[1] Rodriguez, G., and K. Kreutz, "Closed Chain Forward Dynamics by Manipulator Mass Matrix Factorization and Inversion", *JPL Publication*, in preparation, NASA/Jet Propulsion Laboratory, California Institute of Technology, 1988.

[2] Milman, M., K. Kreutz and A. Lokshin, "Coordination and Load Balancing for Multi–Arm Systems," Int. Symp. on Robotics, Albuquerque, NM, Nov., 1988.

[3] Nakamura, Y., K. Nagai and T. Yoshikawa, "Mechanics of Coordinative Manipulation by Multiple Robotic Mechanisms," Proc. 1987 IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, March, 1987. pp. 991–998.

[4] Kreutz, K. and J.T. Wen, "Attitude Control of an Object Commonly held by Multiple Robot Arms: A Lyapunov Approach," Proc. 1988 American Control Conference, Atlanta, GA, June, 1988.

[5] Wen, J.T. and K.Kreutz, "Globally Stable Control Laws for the Attitude Control Problem: Tracking Control and Adaptive Control," IEEE Dec. and Control Conference, Austin, TX, 1988.

[6] Kreutz, K. and A. Lokshin "Load Balancing and Control of Closed Chain Multiple Arm Systems: The Rigid Grasp Case," Proc. 1988 American Control Conference, Atlanta, GA, June, 1988.

[7] Takegaki, M. and S. Arimoto, "A New Feedback Method for Dynamic Control of Manipulators," ASME J. Dynamic Systems, Measurement and Control, **102**, June, 1981.

[8] Wen, J.T., "First Phase Motion/Force Control Strategies for Multiple Robot Arms Holding a Common Rigid Object," JPL EM # 347–88–241 (internal document), June, 1988.

[9] Wen, J.T., and K. Kreutz, "Stability Analysis of Multiple Rigid Robot Manipulators Holding a Common Rigid Object", IEEE Dec. and Control Conference, Austin, TX, 1988.

[10] Popov, V.M., *Hyperstability of Control Systems*, Springer–Verlag, New York, 1973.

[11] Yuan, J.S.C. and W.M. Wohnam, "Probing Signals for Model Reference Identification," IEEE Trans. on Automatic Control, AC–22, No.4, pp.530–538.

[12] Wen, J.T. and D.S. Bayard, "A New Class of Control Laws for Robotic Manipulators, Part I: Non–Adaptive Case," Int. J. Control, **47**, 5, 1988. pp. 1361–1385.

[13] Desoer, C.A. and M. Vidyasagar, *Feedback Systems: Input–Output Properties*, Academic Press, New York, 1975.

Experiments in Cooperative Manipulation: A System Perspective

Stanley A. Schneider
Robert H. Cannon Jr.

Stanford University Aerospace Robotics Laboratory
Stanford, California 94305

Abstract

This paper outlines an experiment in cooperative robotic manipulation conducted at Stanford's Aerospace Robotics Laboratory. In addition to cooperative dynamic control, the system incorporates real-time vision feedback, a novel programming technique, and a graphical high-level user interface. By focusing on the vertical integration problem, we are examining not only these subsystems, but also their interfaces and interactions.

The control system implements a multi-level hierarchical structure; the techniques developed for operator input, strategic command, and cooperative dynamic control are presented. At the highest level, a mouse-based graphical user interface allows an operator to direct the activities of the system. Strategic command is provided by a table-driven finite state machine; this methodology provides a powerful yet flexible technique for managing the concurrent system interactions. The dynamic controller implements "object impedance control"—an extension of Nevill Hogan's impedance control concept to cooperative-arm manipulation of a single object.

Experimental results are presented, showing the system locating and identifying a moving object, "catching" it, and performing a simple cooperative assembly. Results from dynamic control experiments are also presented, showing the controller's excellent dynamic trajectory tracking performance, while also permitting control of environmental contact forces.

1 Introduction

This paper presents an overview of the Dynamic and Strategic Control of Co-Operating Manipulators (DASCCOM) project at Stanford's Aerospace Robotics Laboratory. Due to space constraints, this paper can only present a very brief overview of the system capabilities; more technical detail and experimental results can be found in [9,8,2]. Space considerations also prohibit an extensive literature review; this work draws most heavily on [5,7,3,6]; related research can be found in [1,4,11].

Research goals Space construction requires the manipulation of large, delicate objects. Single manipulator arms are incapable of quickly maneuvering these objects without exerting large local torques. Multiple cooperating arms do not suffer from this limitation. Unfortunately, utilizing multiple manipulators introduces many additional problems, among them dynamic complexity and difficult strategic command.

The goal of this project is to study simultaneously the dynamic and strategic issues of cooperative manipulation, and to demonstrate experimentally a cooperative robotic assembly. We aim not only to master the dynamic control problem, but also to provide for simple, conceptual *direction* of motion by an untrained operator.

Summary of results The DASCCOM project is now essentially complete, and is capable of performing simple assembly operations. Implementation of a complete multiple-robot hierarchy has resulted in several new insights into manipulation and interacting real-time systems. This system integrates for the first time:

- An "object-only" task-specification graphical user interface.
- Finite state table programming, a structured technique for managing asynchronous real-time events and interacting processes.
- Object impedance control, a cooperative dynamic controller that enforces a specified *object behavior*.
- A real-time "point-tracking" vision system, capable of identifying and tracking multiple objects.

2 Experimental Dual Manipulator System

The experimental system is designed to emulate a dual-armed space robotic vehicle. Dual two-link robotic arms (fixed-base in these initial experiments) can manipulate small, freely-floating air-cushion vehicles. The vehicles are equipped with connectors that can mate with several docking ports in the manipulators' workspace. The system thus simulates, in two dimensions, the weightless space manipulation and assembly problem.

Mechanical Hardware The experimental facility consists of a pair of two-link manipulators, affixed to the side of a "small" granite table (4 feet × 8 feet). Each arm is of the popular SCARA configuration—basically anthropomorphic, with vertical-axis, revolute "shoulder" and "elbow" joints. The arms are equipped with joint angle sensors and endpoint force sensors. An overhead television camera provides global vision. A photograph of the experimental setup appears in Figure 1.

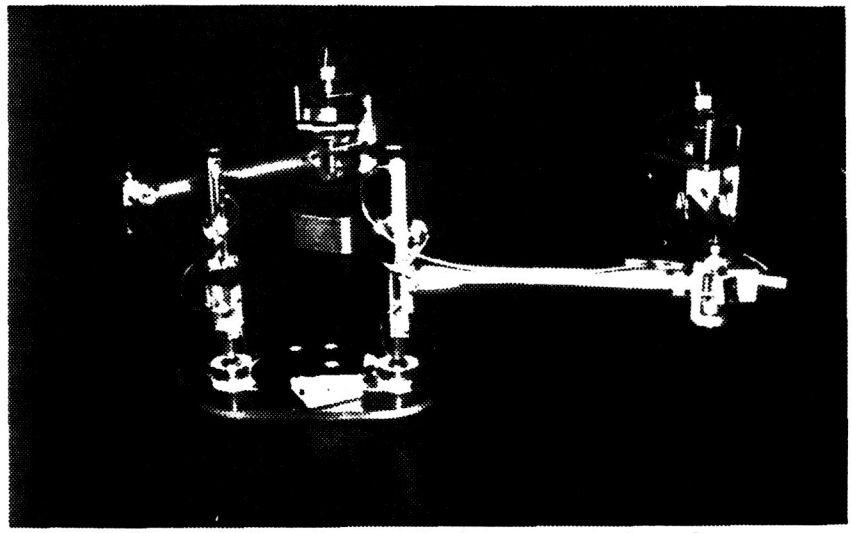


Figure 1: Experimental Dual Arm Manipulator System

Computer System Our real-time computer system combines a proven UNIX development environment with high-performance real-time processing hardware. Motorola 68020/68881 single board processors running the pSOS real-time kernel provide inexpensive real-time processing power. VME bus shared-memory communications permit efficient multiprocessor operation. The real-time processors are linked, via the VME bus, to our Sun/3 engineering workstations. Thus, we benefit from Sun's superb programming environment, while providing the capacity for relatively cheap, unlimited processing expansion.

Real-time software environment Each real-time processor runs pSOS; a small, fast, priority-driven multi-tasking kernel [10]. The features used most heavily by our software structure are the multi-tasking scheduler, the inter-process message facility, and the event-signal facility. We have also developed a large array of in-house real-time software to support control-systems research, [8] presents details.

3 System Structure

The cooperating arms application software consists of four major modules: user interface, strategic control, dynamic control, and vision. The dynamic control module is further divided into three sub-modules: the object impedance controller and dynamic controllers for each of the two arms (see Figure 2). Execution of these modules is spread over three real-time 68020 processors and the Sun workstation.

The command hierarchy The user interface collects conceptual-level commands from the operator, and communicates them to the strategic controller.

The strategic control module is responsible for the overall command of the system. It fields high-level requests from the user interface module, and translates them into sequences of primitives that the dynamic control module can implement. It also monitors the various conditions and activities of the system and directs appropriate response actions.

The dynamic control module is responsible for reading the various sensors and calculating the actuator torques required to produce the desired system behavior.

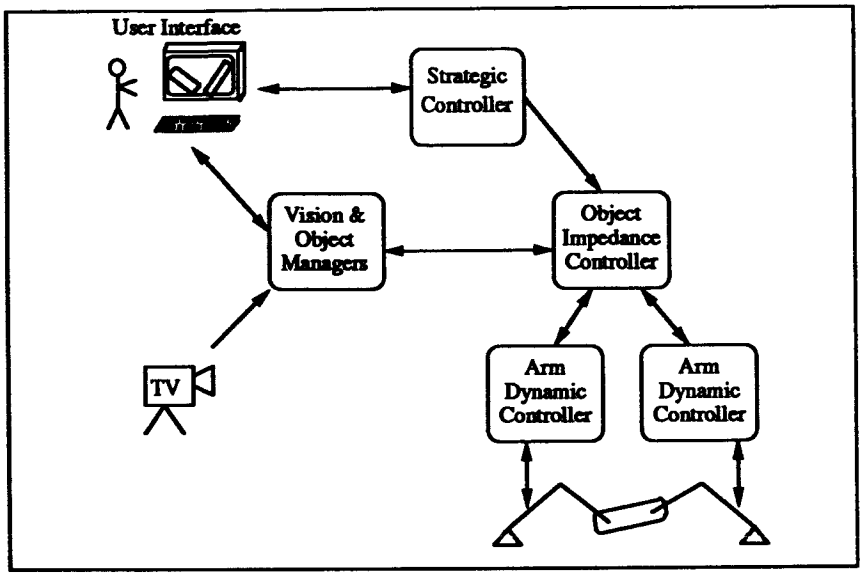


Figure 2: System Structure

System data-flow Data communications between modules is accomplished via simple global data structures. An interprocessor lock gate is provided with each structure. Two major databases are maintained: the "object" database contains physical properties and current states of the objects in the workspace (world model), and the "arms" database contains the desired and measured states (positions, endpoint forces, etc) of each arm.

Since command or temporal information flow is not efficient with shared data structures, bidirectional byte-stream "channels" are also provided for interprocessor (or interprocess) communications.

4 User Interface

The purpose of the user interface is to gather conceptual commands from the user, and communicate them to the strategic control module. The user interface should present a clear and intuitive means for the user to specify his wishes.

Modes of operation Users often wish to communicate with the system at different conceptual levels. To provide this, the user interface of DASCCOM provides two modes of operation: autonomous execution of common operations, and manual "teleoperated" manipulation for unusual tasks. This combination allows the completion of most assembly tasks.

In automatic mode, two "views" of the object being manipulated are displayed. The actual position of the object is displayed by a solid-lined iconic figure. In addition, a "desired" position of the object is drawn with broken lines. The user can move the broken-line (ghost) object by clicking on it and dragging it around the screen.

At all times, the ghost object represents the state that the system will attempt to produce if the mouse button is released. It is thus a one-move preview of the new state of the system. For instance, to perform a connection operation, the user can simply point to any connector on an object, and drag it to any matching connector in the workspace. The display will show the ghost object in the final connected position. When the move is confirmed, the system performs the sequence of actions required to make the connection, and reports the status back to the user. This allows quick, simple assembly operations.

In recognition of impossibility of anticipating all actions, a manual operation mode is also provided. In manual mode, no ghost object is displayed. Instead, manual mode allows direct access to the object impedance dynamic controller.

Both modes utilize an *object-only* interface; the system takes responsibility for all arm motions.

Interface to the strategic controller The user interface executes on the Sun workstation. It communicates with the strategic control module on the real-time system via a bidirectional byte-stream channel through VME-bus shared memory.

The protocol utilizes a "request/response cycle" pattern; the user interface requests an action, and the strategic controller returns a status response when the action completes. The simplicity of this communication paradigm allows considerable flexibility.

A brief operational description Several frames from a typical session are presented as Figure 3. The screen is divided into three sections. The large lower section depicts the manipulator workspace in iconic form; the activities of the system are visually displayed here. The upper left window displays the system status; the first line in this window gives a short verbal description of the systems activity. A system control panel forms the upper right section.

In the upper right (second) frame, for example, there are two objects in the vision system's field of view. Scooter is the floating air-cushion object. Scooter has two gripper attachment ports and two male connectors. Multibase is a stationary object with female connectors. The arms are currently holding Scooter, as evidenced by the presence of the Scooter ghost image. The user has just dragged the ghost's right connector over to Multibase's rightmost connector. The status line indicates that the insertion of one of Scooter's connectors into one of Multibase's connectors is in progress.

Note that the arms do *not* appear in the display. The operator commands only object motions; the arm actions required to effect these motions need not be specified.

An example task: install a part For the sake of this example, suppose that "Multibase" is affixed to a mobile robot, and represents a series of attach points for holding miscellaneous items. "Scooter" is a part that is to be installed into a remote module, represented by "Dock".

In the upper left frame, the part is approaching the robot system[1]. The operator has just indicated Scooter is to be grasped, thus the "Acquiring Scooter" status message in the top left corner. In the following frame, the operator indicates that Scooter should be affixed to the robot's base. Next, the robot is directed to navigate to the vicinity of Dock (navigational control is not discussed here). When Dock is in view, the operator indicates that the new part (Scooter) is to be installed. The lower left frame shows the first stage of that action. Finally, in the last frame, the part has been released, and the installation is complete.

This entire procedure was accomplished with only four simple mouse motions. (Click on the approaching Scooter, connect Scooter to Multibase, connect Scooter to Dock, release.) Most of the assembly details—such as how to attach and detach connectors, how fast to approach the docking connector, etc.—are completely automated.

Summary: User interface In summary, the DASCCOM user interface features:

- A simple mouse-based "point and click" graphical interface.
- "Object-only" task specification; manipulation details are left to the system.
- One-step operation previewing.
- Automatic execution of most assembly operations.
- Optional "manual" manipulation at the dynamic object control level.

[1] Or, equivalently, the robot is approaching the part.

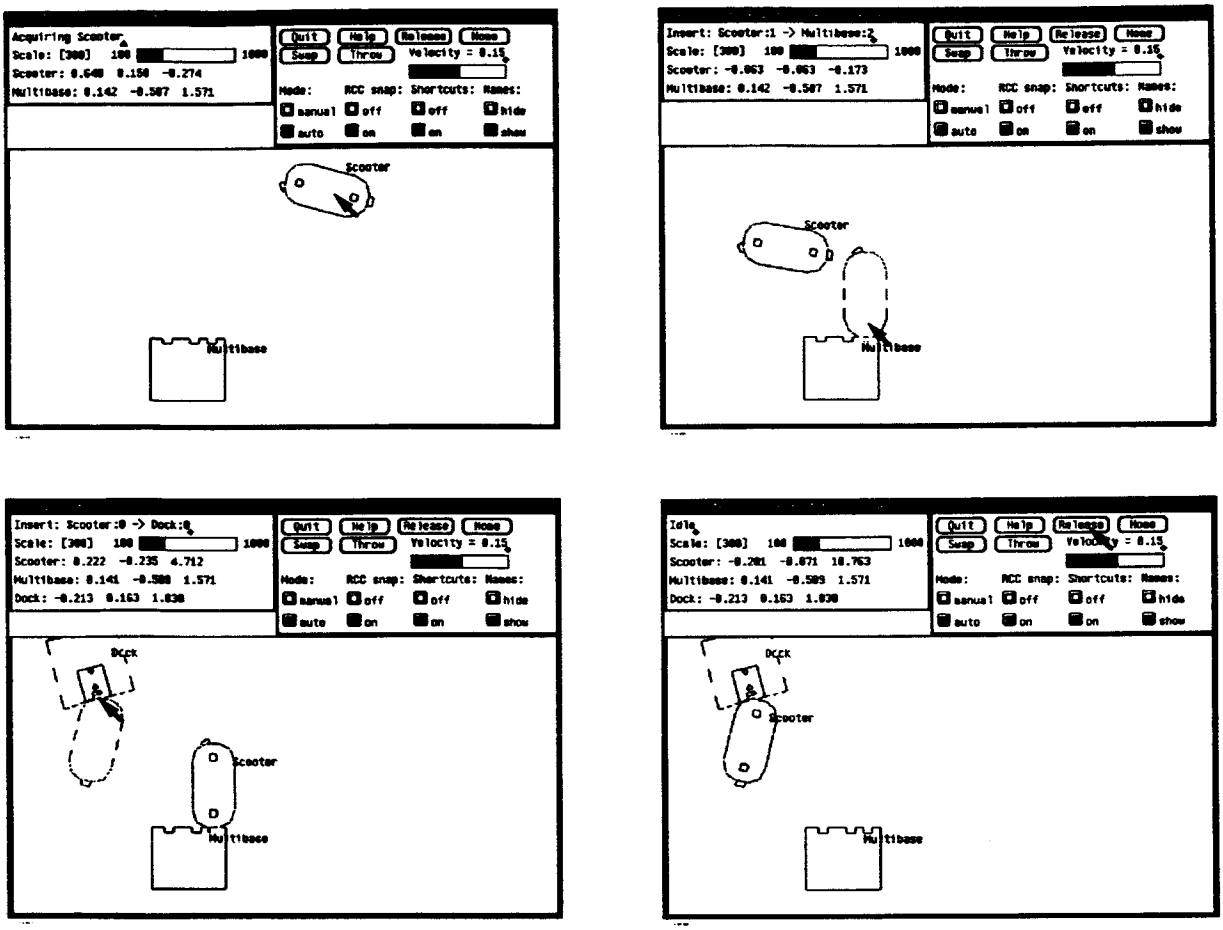


Figure 3: Installation Demonstration Example

5 Strategic Control

The purpose of the strategic controller is to provide an interface between the conceptual commands provided by the user interface and the dynamic control module. It furnishes the user interface with a set of simple automated commands, allowing it to perform many tasks.

An integrated system involving multiple manipulators, real-time vision, and interactive operator control is a complex, event-driven environment. These systems are naturally concurrent in nature; complex asynchronous interactions must be managed. With a fundamentally sequential underlying programming paradigm, the burden of managing these asynchronous events is left to the programmer. A sequential execution stream model can not, for instance, deal effectively with multiple-arm synchronization, especially if the arms are running different programs on different processors.

The state table programming technique This section presents an alternative programming methodology, referred to as state table programming. It provides a naturally event-driven structure to guide the programmer in producing code that is easily interfaced to other real-time system modules. It directly exploits the facile multiple-process generation and communications provided by modern real-time kernels. In fact, management of multiple asynchronous events is central to the structure of the system; the programmer is actively encouraged to divide the problem into small, independently executing programs. In addition, it is based on a very intuitive task description technique.

The state table programming technique is neither a robot programming language nor a replacement for a library of robot control routines. For instance, trajectory generation utilities and world modeling utilities are also required. The technique merely provides a framework that weaves the multiple streams of execution in the system into an easy-to-use structure.

State transition graphs State transition graphs provide a simple, intuitive means of visualizing the sequence of actions required to effect a task. A "state" is usually characterized by the system performing a single "step" of an operation, and waiting for some indication of its completion. State transitions are indicated by arrows labeled with the event that causes the transition. When an event occurs, a transition routine is executed. The result of that routine determines the next state entered.

For example, Figure 4 presents a simplified series of steps required to catch a moving object. The system starts out in the "Idle" state. The receipt of the "Acquire" stimulus, presumably sent from a higher level (e.g. the user interface), causes the system to enter one of two states: "Reaching" if the object is within reach, or "Waiting", if not. While the system is in the "Reaching" state, the arms are executing an intercept trajectory. When the trajectory completes ("TrajComplete" event) , the arms track the object until the gripper endpoints match the targeted grip ports precisely. At this point, the system enters the "Gripping" state while the grippers engage. Finally, the catch is complete, and the "Manipulating" state is active until a "Release" command is received.

This is a rather simple example. Much more complex series of actions are easily representable.

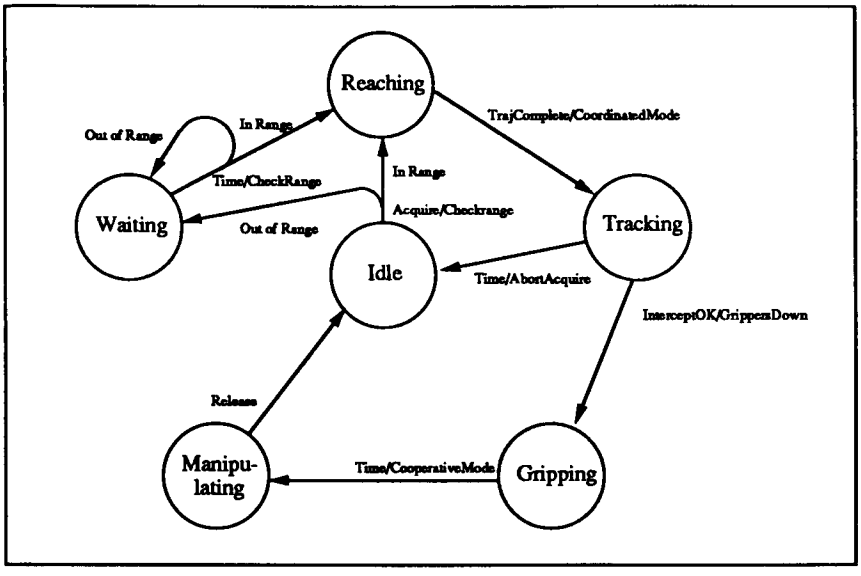


Figure 4: State Transition Graph - Catch Task

State table programming The state table entries corresponding to the states of Figure 4 are presented in table 1. Note that multiple pending conditions are handled very naturally, for example, the "Tracking" state waits for either "InterceptOK" or "Time".

State	Stimulus	Transition Routine	Next States	
Idle	Acquire	CheckRange	Reaching	Waiting
Waiting	Time	CheckRange	Reaching	Waiting
Reaching	TrajComplete	CoordinatedMode	Tracking	
Tracking	InterceptOK	GrippersDown	Gripping	
	Time	AbortAcquire	Idle	
Gripping	Time	CooperativeMode	Manipulating	
Manipulating	Release	GrippersUp	Idle	

Table 1: State Transition Table

Stimuli may be generated by any of the modules of the system, but most originate from one of two sources: command stimuli from the user interface, and condition-event stimuli from independently executing "helpers". The "Acquire" and "Release" stimuli are examples of the former. The "Time" stimulus is created by a simple helper process that sleeps for a specified time before sending its message. Other helpers (not shown in the table) perform "safety checks"; for example, the "OutOfRange" stimulus is sent by a process that checks every so often to insure that the object being acquired is not suddenly moved out of range.

Implementation The heart of the system is the Finite State Machine (FSM) driver. The FSM driver is an independently executing process. It acts as a central command post, receiving messages from many sources in the system and taking the appropriate action. Each message contains a stimulus code; the FSM driver uses that code to reference the state table and select a state transition routine to execute. The state transition routines perform the actual work: they change controller modes, start and stop helper processes, and interact with the trajectory generation module.[2]

The DASCCOM strategic control module supports automatic object capture, docking (connector insertion), withdrawal, and throwing functions. Each task is executed as a multi-step chain of state transitions.

Summary: Strategic control The DASCCOM strategic controller is based on an event-driven finite state machine. This technique features:

- An intuitive, graphical task specification.
- Direct tabular implementation.
- A natural, event-driven structure.

6 Cooperative Dynamic Control

The dynamic equations of motion of multiply-armed robotic systems are complex. Strategic control of the system interactions is also difficult; a consistent set of desired motions must be specified. As of yet, no satisfactory method of precise dynamic control coupled with a simple strategic command interface has been developed and experimentally demonstrated.

This section outlines a strategy for the control of a cooperative robotic system that permits high performance dynamic motion control, while also allowing direct control of environmental interactions. This is accomplished by controlling the manipulated *object* to react to external environmental stimuli with a programmable impedance. This facilitates motion direction by presenting a simple yet powerful interface; the strategic controller need only specify the impedance. Although "exact" inertial force compensation is achieved, the control structure does not require explicit formulation of the closed-chain dynamic equations of motion, and is amenable to parallel computation. Object internal forces are explicitly controlled. The object impedance controller has been implemented on a multi-processor real-time computer system. Experimental results are presented in section 8 to verify the controller's performance, both for free-motion slews and environmental contact.

Control objective Hogan's impedance control policy [5] causes the endpoint of the manipulator to react to external forces with a programmable impedance. The simplest example both to understand and implement is a simple second order linear impedance—the endpoint behaves as a mass attached via a virtual spring-damper to the environment. DASCCOM utilizes a dynamic controller that enforces a controlled impedance not of the arm *endpoints*, but of the manipulated *object* itself. Intuitively, the object behaves as if it were attached to its environment by linear spring-damper systems in the linear degrees of freedom, and also by uncoupled torsional spring-dampers to control rotational orientation.

The object impedance controller enforces (for a simple linear second-order impedance) the relationship:

$$m_d(\ddot{x} - \ddot{x}_{des}) + k_v(\dot{x} - \dot{x}_{des}) + k_p(x - x_{des}) = f_{ext}$$

Here x denotes the coordinate of any one degree of freedom of an *arbitrary* point fixed in the object's frame. The constants m_d, k_v, and k_p are specifiable. The reference signal x_{des} denotes the desired position (or orientation) of the chosen point. The $\ddot{x}_{des}$ term represents acceleration feed-forward. Thus, the programmable impedance force corrects deviations from the desired trajectory.

Intuitively, this control policy completely supplants the actual dynamics of the object with a "virtual" object, with specifiable mass and inertia properties[3]. The "virtual" object is attached at its (apparent) center of mass via an orthogonal set of imaginary damped springs to a selectable point in the environment. Thus,

[2] The addition of callable sub-chains would add considerable power to the implementation; it is under development.

[3] Of course, this can only be done within the bandwidth and actuation limits of the system.

the object can be manipulated by simply moving the virtual spring endpoint. Controlled force interactions with environmental obstacles can be done by simply pressing the "spring" against the obstacle. Thus, both free motion slews and manipulation requiring contact can be done with the same strategic interface.

The position and orientation of the "virtual" object with respect to the actual object is also selectable. This selectable "command frame" allows simple specification of many operations. A particularly useful example is for performing connector insertions—by placing the "virtual" object frame at a fixed location in the connector frame, all assembly operations can be specified as connector motions only. Multiple connectors arranged on an object in arbitrary orientations can then be handled by the same simple connector insertion algorithm. Since the stiffness is selectable, this controller is also capable of pseudo remote center of compliance (RCC) operation [12], permitting simple and efficient part mating and insertion operations.

Summary: Dynamic Control Space constraints prohibit derivation of the controller here, see [9] for a more complete treatment. The controller features:

- A simple, powerful *object behavior* specification interface.
- Good dynamic performance, both in free motion and in contact, without switching controllers.
- Exact dynamic compensation, without requiring closed-chain equations of motion.
- A selectable command frame, facilitating assembly operations.

7 Real-time Vision System

To track moving objects, DASCCOM employs a high-speed television camera-based point-tracking vision system. The vision system uses a 60 Hz shuttered CCD television camera to track special variable reflectivity targets. Each object to be tracked is outfitted with a unique pattern of these targets. The vision system software is then able identify and track individual objects in real-time. Simple linear state estimators also provide velocity estimates. The arm endpoints are also fitted with targets, and the information is used to allow endpoint feedback control.

Space considerations prohibit further description here; the system is described in detail in [2]. In summary, the vision system capabilities are:

- Real-time (60Hz) *object* position and orientation.
- Sub-pixel resolution (about $1/20^{th}$ of a pixel).
- Multiple object identification and tracking.

8 Experimental Results

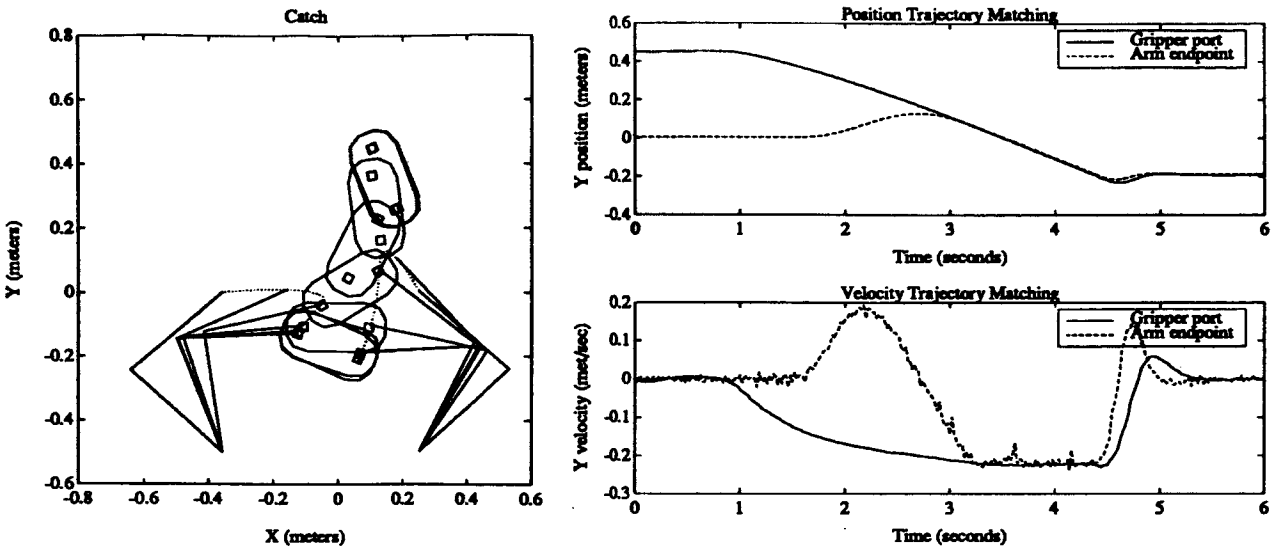
A moving object "catch" A "strobe" sequence picture of a typical catch is presented as the left side of Figure 5. The vision system provides high-speed data for three subsystems to perform this task: the two arms, and the moving body. Each arm is under vision-guided endpoint control. The desired trajectory for each arm takes it from its initial "home" position, to an intercept state that matches the object's gripper port in both position and velocity.

To perform a successful capture, the arm endpoint must then be held over the gripper port for the duration of the time required for the gripper mechanism to engage. The positioning must be fairly accurate.

The right side of Figure 5 shows the vertical positions and velocities of the right arm and right gripper port during the catch. The object is being accelerated toward the arm system for the first second. During the next second, the arm tip accelerates to match the port position and velocity at about the 3.2 second mark[4]. The gripper's downward motion occupies the next second, after which the object is brought to a halt. After the grippers have engaged (at about 4.4 second mark) the observer has an incorrect plant model; this accounts for the apparent difference in position and velocity after the capture.

Before the grippers engage, they are under independent endpoint impedance control; after the object is caught the object impedance controller is in effect. Thus, compliant control is active at all times—this prevents large acceleration forces during and after the acquisition.

[4]The arm trajectories are actually updated several times as the object position and velocity estimates improve.



Overhead View Trajectory Matching

Figure 5: A Two-handed Catch

Transport trajectory tracking Five cooperative control algorithms were compared for a cooperative transport task. The five algorithms are: co-located proportional-derivative (PD) joint-space coordinated position control, dynamic and kinematic coordinated endpoint impedance control, and dynamic and kinematic object impedance control. Space constraints only allow presentation of results from two of the controllers here; see [9,2,8] for more details. The commanded reference is a fifth-order trajectory of the center of mass of the object in each object degree of freedom: x, y, and θ. All algorithms were provided with the correct coordinated position, velocity, and acceleration references for the entire slew path.

Figure 6 compares the PD controller to the dynamic object impedance controller. The upper-left plot for each controller depicts the motion of the center of mass of the y direction. The lower-left is the corresponding velocity. The upper-right plots x vs. y, and indicates the desired and actual object positions at 0.5 second intervals during the motion. The lower-right plot shows the magnitude of the "tension" between the arms, after being corrected for dynamic forces. Both controllers are attempting to maintain zero tension.

The PD controller does a poor job of following the desired trajectory and offers no control of the internal forces on the object. This controller also does not compensate for inertial forces.

Since the object impedance controller correctly compensates for the object dynamics, the trajectory tracking performance is quite impressive. The inter-arm tension is controlled well also.

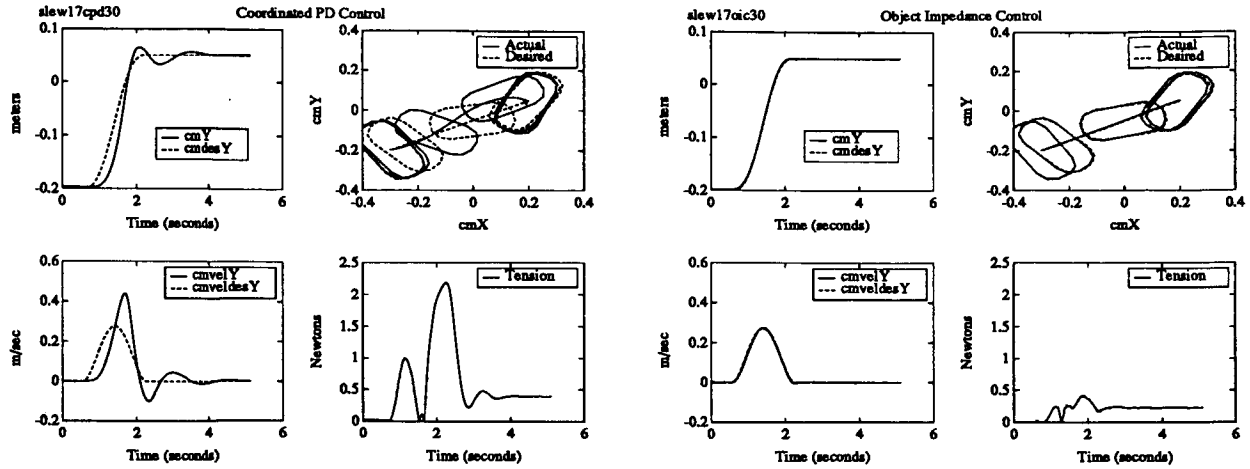
Force control performance Force control data (Figure 7) were obtained by simply placing a hard, stationary object in the path of the object during a slew. The unfiltered data exhibits some ringing—this is an impact between two very hard objects—but the force level quickly settles to the desired. The important thing to note is that the object impedance controller successfully controls the forces of interaction, without switching control modes, even when it comes into contact with a very stiff environment.

9 Conclusions

This paper has presented an overview of the DASCCOM system. This system integrates strategic and dynamic control of cooperating manipulators with a real-time vision system and a graphical user interface. The techniques developed have been experimentally proven, and will provide a basis for the Stanford Aerospace Robotics Laboratory's future space robotics research.

References

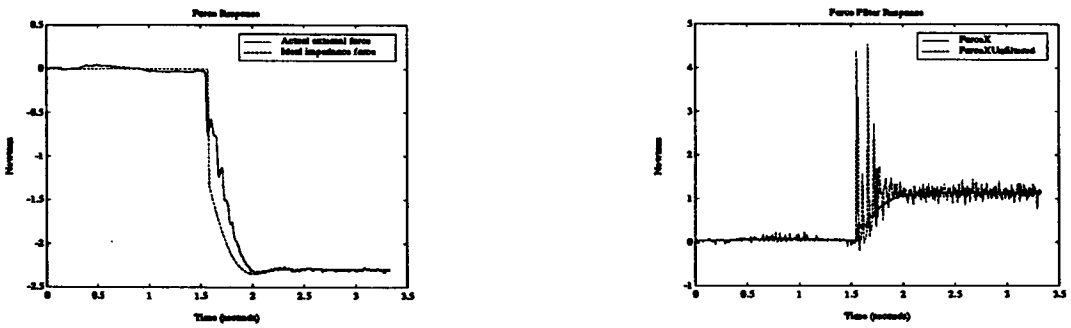
[1] T. E. Alberts and D. I. Soloway. Force control of a multi-arm robot system. In *Proceedings of the International Conference on Robotics and Automation*, pages 1490 – 1496, Philadelphia, PA, April 1988. IEEE.



Proportional-Derivitive

Object Impedance Control

Figure 6: Controller Slew Performance Comparison



Estimated External Force

Right Arm Force Sensor Output

Figure 7: Impact Force Control Performance

[2] R. H. Cannon, Jr., M. Ullman, R. Koningstein, S. Schneider, W. Jasper, R. Zanutta, and B. Dickson. NASA Semi-Annual Report on Control of Free-Flying Space Robot Manipulator Systems. Semi-Annual Report 7, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, August 1988.

[3] S. Hayati. Hybrid position/force control of multi-arm cooperating robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 82 – 89, San Francisco, CA, April 1986. IEEE.

[4] G. Hirzinger and J. Dietrich. Multisensory robots and sensor-based path generation. In *Proceedings of the International Conference on Robotics and Automation*, pages 1992 – 2001, San Francisco, CA, April 1986. IEEE.

[5] N. Hogan. Impedance control: An approach to manipulation, parts i, ii and iii. *Trans of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, March 1985.

[6] O. Khatib. Object manipulation in a multi-effector robot system. In *ISRR*, Santa Cruz, CA, 1987.

[7] Y. Nakamura, K. Nagai, and T. Yoshikawa. Mechanics of coordinative manipulation by multiple robotic mechanisms. In *Proceedings of the International Conference on Robotics and Automation*, pages 991 – 998, Raleigh, NC, April 1987. IEEE.

[8] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, May 1989.

[9] S. Schneider and R. H. Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. In *Proceedings of the International Conference on Robotics and Automation*, Tempe, AZ, April 1989. IEEE.

[10] Software Components Group, Inc., 4655 Old Ironsides Drive, Santa Clara, CA 95054. *pSOS - 68K Real-Time, Multiprocessing Operating System Kernel User's Manual*, 4.1 edition, December 1986.

[11] M. Uchiyama, N. Iwasawa, and K. Hakomori. Hybrid position/force control for coordination of a two-arm robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 1242 – 1247, Raleigh, NC, April 1987. IEEE.

[12] D. E. Whitney and J. L. Nevins. What is the remote centre compliance (rcc) and what can it do? *Robot Sensors*, 2:3–15, 1986.

ON THE MANIPULABILITY OF DUAL COOPERATIVE ROBOTS

P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano

Dipartimento di Informatica e Sistemistica
Università di Napoli
Via Claudio 21, 80125 Napoli, Italy

Abstract

In this paper the definition of manipulability ellipsoids for dual robot systems is given. A suitable kineto-static formulation for dual cooperative robots is adopted which allows for a global task space description of external and internal forces as well as absolute and relative velocities. The well-known concepts of force and velocity manipulability ellipsoids for a single robot are formally extended and the contributions of the two single robots to the cooperative system ellipsoids are evidenced. Duality properties are discussed. A practical case study is developed.

1. Introduction

Cooperative robots have been recognized by the robotics research community as offering enhanced capabilities over current single robot structures. Dual robot cooperation allows for performing tasks such as handling large, heavy and non-rigid objects, assembly and mating mechanical parts, which could not be executed by a single robot. Another advantage is the enlargement of the reachable workspace. All the above features play a crucial role, for instance, in space robotics applications where cooperative manipulation is often considered as an essential requirement.

In spite of the potential benefits achievable with dual robots, the control problem becomes more complex due to the kinematic and dynamic interactions. A must for the solution of this kind of problem is constituted by an effective description of the kineto-static and dynamic relationship for a general dual robot system. To this purpose, the formulation proposed by Dauchez and Uchiyama [1] has been shown to be suitable to coordinated control schemes with equal importance attributed to the two robots performing a given task. Their approach is somewhat opposite to the master-slave strategy suggested by Luh and Zheng [2] which has been argued by Uchiyama et al. [3] to be unsuitable for practical position/force control of dual robots.

It is believed that an important issue is the definition of quantitative measures of the enhanced performance offered by dual robot cooperation. It is well-known that the manipulability ellipsoids introduced by Yoshikawa [4] represent one of such measures for a single robot. The contribution of this work is to provide a systematic way of extending the above concept [4] to the dual robot case. In order to accomplish this goal, the formulation dictated in [3] is followed here. The motivation behind this choice is that it leads to a natural, straightforward derivation of manipulability measures which be consistent with those proposed in [4] and susceptible of an immediate physical interpretation for the closed-chain system created by the dual robots tightly handling an object. A similar, parallel research effort has recently been produced by Lee and Bejczy [5], although the definition of manipulability measures for a dual robot system is obtained according to different criteria related to the effect of one robot on the other, instead of regarding the closed-chain as a whole.

A practical case study is worked out for two simple planar cooperative robots. Velocity and force static ellipsoids are obtained which show the correctness and functionality of the proposed approach.

2. Kineto-static formulation for two cooperative robots

In the following the formulation of task space coordinates required for describing cooperative tasks is briefly summarized from [2]. For the purpose of the present work, the case when the two robots are rigidly attached to the object is considered, i.e. a rigid grasp.

According to [2], the cooperative task is described in terms of a set of absolute coordinates and a set of relative coordinates. The static relationship between the generalized forces exerted by the two robots and the generalized forces acting on the object — external and internal — is presented first. The kinematic relationship will be derived by using the duality relation between forces and velocities. Fig. 1 illustrates two cooperative robots tightly grasping an object. Let m be the common dimension of the task spaces of the two robots and

$$\mathbf{h}_1 = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{n}_1 \end{pmatrix} \qquad \mathbf{h}_2 = \begin{pmatrix} \mathbf{f}_2 \\ \mathbf{n}_2 \end{pmatrix} \tag{1}$$

denote the vectors of the generalized forces (forces $\mathbf{f}_1$, $\mathbf{f}_2$ and moments $\mathbf{n}_1$, $\mathbf{n}_2$) exerted by the two end-effectors, respectively. Let then

$$\mathbf{h}_a = \begin{pmatrix} \mathbf{f}_a \\ \mathbf{n}_a \end{pmatrix} \qquad \mathbf{h}_r = \begin{pmatrix} \mathbf{f}_r \\ \mathbf{n}_r \end{pmatrix} \tag{2}$$

denote the vectors of external and internal forces/moments acting on the object, respectively. Let be

$$\mathbf{f} = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{pmatrix}. \tag{3}$$

It can be shown that

$$\mathbf{h}_a = W\mathbf{f} \tag{4}$$

where

$$W = \begin{pmatrix} I & 0 & I & 0 \\ -R_{1a} & I & -R_{2a} & I \end{pmatrix} \tag{5}$$

with R_{1a}, R_{2a} defined by $\mathbf{f}_1 \times \mathbf{r}_{1a} = -R_{1a}\mathbf{f}_1$, $\mathbf{f}_2 \times \mathbf{r}_{2a} = -R_{2a}\mathbf{f}_2$, respectively. I and 0 denote identity and null matrices of appropriate dimensions. Also it is

$$\mathbf{f} = V\mathbf{h}_r \tag{6}$$

where

$$V = \begin{pmatrix} I & 0 \\ R_{1a} & I \\ -I & 0 \\ -R_{2a} & -I \end{pmatrix}. \tag{7}$$

It can be recognized that the mapping V in (7) spans the null space of the mapping W in (5). This means that the external and internal force/moment vectors belong to orthogonal subspaces [6].

Once the static relationship has been established, the differential kinematic relationship is derived in a similar manner. Let

$$\dot{\mathbf{y}}_1 = \begin{pmatrix} \dot{\mathbf{x}}_1 \\ \omega_1 \end{pmatrix} \qquad \dot{\mathbf{y}}_2 = \begin{pmatrix} \dot{\mathbf{x}}_2 \\ \omega_2 \end{pmatrix} \tag{8}$$

denote the vectors of the velocities (translational $\dot{\mathbf{x}}_1$, $\dot{\mathbf{x}}_2$ and rotational ω_1, ω_2) at the two end-effectors, respectively. Let then

$$\dot{\mathbf{y}}_a = \begin{pmatrix} \dot{\mathbf{x}}_a \\ \omega_a \end{pmatrix} \qquad \dot{\mathbf{y}}_r = \begin{pmatrix} \dot{\mathbf{x}}_r \\ \omega_r \end{pmatrix} \tag{9}$$

denote the vectors of external (absolute) and internal (relative) velocities, respectively. Let be

$$\dot{\mathbf{z}} = \begin{pmatrix} \dot{\mathbf{y}}_1 \\ \dot{\mathbf{y}}_2 \end{pmatrix}. \tag{10}$$

In force of the duality relation between forces and velocities which is derived from the principle of virtual work in mechanics, it can be shown that

$$\dot{\mathbf{z}} = W^T \dot{\mathbf{y}}_a \tag{11}$$

and

$$\dot{\mathbf{y}}_r = V^T \dot{\mathbf{z}} \tag{12}$$

with W and V defined in (5) and (7), respectively.

3. Definition of manipulability ellipsoids

The idea of measuring the manipulating ability of robotic mechanisms was first introduced in [4]. According to that concept, a force manipulability ellipsoid and a velocity manipulability ellipsoid can be defined for a single robot. Assume that an n-DOF robot is given and an m-dimensional task space is of interest, usually with $m \leq n$. It is well-known that

$$\tau = J^T(\theta)\gamma \tag{13}$$

represents the static relationship between the task force vector γ and the joint torque vector τ through the transpose of the Jacobian matrix $J(\theta)$, with θ denoting the joint displacement vector. Dually,

$$\mathbf{v} = J(\theta)\dot{\theta} \tag{14}$$

represents the kinematic relationship between the joint velocity vector $\dot{\theta}$ and the task velocity vector $\mathbf{v}$ through the Jacobian $J(\theta)$.

The unit sphere in the joint torque space

$$\tau^T \tau = 1 \tag{15}$$

maps into the task force space ellipsoid

$$\gamma^T (JJ^T)\gamma = 1 \tag{16}$$

which is called *force manipulability ellipsoid* [4]. Dually, the unit sphere in the joint velocity space

$$\dot{\theta}^T \dot{\theta} = 1 \tag{17}$$

maps into the task velocity space ellipsoid

$$\mathbf{v}^T (JJ^T)^{-1} \mathbf{v} = 1 \tag{18}$$

which is called *velocity manipulability ellipsoid* [4]. Note that the explicit dependence on θ has been dropped in J. A direct comparison of (16) with (18) indicates that the principal axes (related to the eigenvectors) of the two ellipsoids coincide, whilst the lengths of the axes (related to the eigenvalues) are in inverse proportion. This inverse velocity-force relation is consistent with regarding the manipulator as a mechanical transformer [7]. Conservation of energy dictates that amplification

in velocity transmission must invariably be accompanied by reduction in force transmission, and vice-versa.

In the following the concepts of force and velocity ellipsoids defined in (16) and (18) are formally extended to a two robot system, based on the kineto-static formulation given in the previous section. Let n_1 and n_2 be the DOF's of the robots, respectively. The static relationship (13) can be written for a two robot system as

$$\mathbf{t} = J_{12}^T \mathbf{f} \tag{19}$$

with $\mathbf{f}$ defined in (3), where

$$\mathbf{t} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} \tag{20}$$

denotes the extended joint torque vector in an $(n_1 + n_2)$-dimensional space, and

$$J_{12} = \begin{pmatrix} J_1 & 0 \\ 0 & J_2 \end{pmatrix} \tag{21}$$

denotes the extended Jacobian matrix. Solving eq. (19) for $\mathbf{f}$ yields

$$\mathbf{f} = J_{12}^{T\dagger} \mathbf{t} \tag{22}$$

where the simbol "$\dagger$" denotes a pseudo-inverse of proper dimensions; in this case it is a left pseudo-inverse.

The external force manipulability ellipsoid and the absolute velocity manipulability ellipsoid are derived first. Plugging (22) into (4) gives

$$\mathbf{h}_a = J_a^{T\dagger} \mathbf{t} \tag{23}$$

which expresses the relationship between the extended joint torque vector and the external force vector, through the matrix

$$J_a^{T\dagger} \triangleq W J_{12}^{T\dagger} \tag{24}$$

which is analogous to the pseudo-inverse of the Jacobian J^T in (13) for a single robot. At this point the formal definition of the external force manipulability ellipsoid can be given. The unit sphere in the extended joint torque space

$$\mathbf{t}^T \mathbf{t} = 1 \tag{25}$$

maps into

$$\mathbf{h}_a^T (J_a J_a^T) \mathbf{h}_a = 1 \tag{26}$$

which is defined here as *external force manipulability ellipsoid*. Dually, the formal definition of the absolute velocity manipulability ellipsoid can be given. The unit sphere in the extended joint velocity space

$$\dot{\mathbf{q}}^T \dot{\mathbf{q}} = 1 \tag{27}$$

maps into

$$\dot{\mathbf{y}}_a^T (J_a J_a^T)^{-1} \dot{\mathbf{y}}_a = 1 \tag{28}$$

which is defined here as *absolute velocity manipulability ellipsoid*. Notice that in (27)

$$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \tag{29}$$

indicates the extended joint vector.

An attractive mathematical expression can be found for the matrix $J_a J_a^T$ constituting the core of the two manipulability ellipsoids just defined in (26) and (28), which is directly related to the Jacobians of the two robots defined in (21). As shown in Appendix A, one may obtain

$$J_a J_a^T = ((\tilde{J}_1 \tilde{J}_1^T)^{-1} + (\tilde{J}_2 \tilde{J}_2^T)^{-1})^{-1} \tag{30}$$

with

$$\tilde{J}_i^{T\dagger} = J_i^{T\dagger} + \begin{pmatrix} 0 \\ -R_{ia}(J_i^{T\dagger})_f \end{pmatrix} \qquad i = 1,2 \tag{31}$$

where the subscript f refers to the upper block of the matrix $J_i^{T\dagger}$ which maps the joint torque vector τ_i into the sole force components of the task force vector $\mathbf{f}_i$ defined in (1) (i.e. excluding the moment components). It is worth noticing that in the particular case when $R_{ia} = 0$, the matrices $\tilde{J}_i^{T\dagger}$ simplify to $J_i^{T\dagger}$.

In the same formal manner as done above for the external force manipulability ellipsoid and the absolute velocity manipulability ellipsoid, the derivation of the internal force manipulability ellipsoid and the relative velocity manipulability ellipsoid is presented now. Plugging (22) into (6) gives

$$\mathbf{t} = J_r^T \mathbf{h}_r \tag{32}$$

with

$$J_r^T \triangleq J_{12}^T V. \tag{33}$$

It is to be remarked that, by virtue of the definitions (24) and (33) and of the structure of the matrices W and V, it results

$$J_a^{T\dagger} J_r^T = 0. \tag{34}$$

The unit sphere (25) maps into

$$\mathbf{h}_r^T (J_r J_r^T) \mathbf{h}_r = 1 \tag{35}$$

which is defined as *internal force manipulability ellipsoid*. Dually, the unit sphere (27) maps into

$$\dot{\mathbf{y}}_r^T (J_r J_r^T)^{-1} \dot{\mathbf{y}}_r = 1 \tag{36}$$

which is defined as *relative velocity manipulability ellipsoid*. In this case too, an attractive mathematical expression can be found for the matrix $J_r J_r^T$ constituting the core of the two manipulability ellipsoids just defined in (35) and (36), which is directly related to the Jacobians of the two robots defined in (21). As shown in Appendix B, one may obtain

$$J_r J_r^T = \hat{J}_1 \hat{J}_1^T + \hat{J}_2 \hat{J}_2^T \tag{37}$$

with

$$\hat{J}_i^T = (-1)^{i-1} J_i^T + ((-1)^{i-1} (J_i^T)_n R_{1a} \quad 0) \qquad i = 1,2 \tag{38}$$

where the subscript n refers to the lower block of the matrix J_i which maps the sole moment components of the task force vector $\mathbf{f}_i$ defined in (1) into the joint torque vector τ_i.

From eq. (34), it directly follows that

$$\tilde{J}_1^{T\dagger} \hat{J}_1^T - \tilde{J}_2^{T\dagger} \hat{J}_2^T = 0. \tag{39}$$

It is worth noticing that in the particular case when $R_{i_a} = 0$, the matrices $\hat{J}_i^T$ simplify to J_i^T; in this case, thus, eq. (39) trivially holds.

Eqs. (30) and (37) suggest a nice interpretation of the way the Jacobians of the two robots combine to form the respective cores of the ellipsoids defined above. If each term of the type JJ^T is regarded as a generalized impedance, eq. (30) resembles the mathematical expression of the parallel of two impedances, whilst eq. (37) resembles that of the series of two impedances. Therefore, one would naturally be driven to generalize these results to the multiple robot case; this topic is under investigation.

4. Case study

Two 3-DOF planar robots are considered for the purpose of illustrating the application of the concepts presented in this work to a practical two robot system. For the sake of simplicity, the end-effectors of the two robots are supposed to be located in the same point (i.e. the physical object is removed); this implies that $\tilde{J}_i = \hat{J}_i = J_i$. This assumption is not restrictive at all, as formally shown above. Moreover, a two-dimensional global task space is assumed, i.e. only forces and linear velocities are of interest; the system thus possesses two redundant DOF's.

A CAD tool has been developed which is articulated into the following steps. The contact point of the two end-effectors is input, then the two redundant DOF's are exploited to assign the orientation angles of the end-effectors. A software package for solving the inverse kinematics of general robot structures [8] is utilized to find the joint configurations and then the complete kineto-static characterization of the system. An option is provided to compute the ellipsoid of interest. The outputs are plotted by means of a graphic package. They illustrate the closed kinematic chain together with the principal axes of the ellipsoids of the two single robots and those of the dual robot system, in order that the manipulability of the cooperative system can be evaluated with respect to the single robot manipulabilities.

Two complete sets of results for two different configurations of the dual robot system are displayed in Figs. 2 and 3 respectively. The ellipsoids of each robot are included for a better comprehension of the effects of the cooperation. It is remarkable that, in the second configuration, the two robots are both proximal to singular configurations (see the shape of their ellipsoids). It can be recognized that the external force ellipsoids (Figs. 2a and 3a) are improved in that the ability of each robot to exert forces along a given direction is enhanced by the other, while the absolute velocity ellipsoids (Figs. 2b and 3b) show that the ability of each robot to perform motions along a given direction is penalized by the presence of the other. This result well agrees with practice, since it is intuitive that when two robots cooperate the static force is shared by them whereas the faster robot is slowed down by the other. Conversely, the ability of the system to absorbe forces along a direction is limited by the weaker robot of the chain (Figs. 2c and 3c), while the ability of the system to give rise to relative motions along a direction is supplied by both robots (Figs. 2d and 3d). All these conclusions reflect the concept of duality which is at the basis of the definition of the manipulability ellipsoids presented.

5. Conclusions

The concept of manipulability ellipsoids has formally been extended to the case of dual robot systems. A global kineto-static formulation of the closed chain created by two tightly cooperating robots has been exploited to define external and internal force manipulability ellipsoids. The corresponding absolute and relative velocity manipulability ellipsoids have been derived on the basis of the duality principle in mechanics. Functional expressions for these ellipsoids have been obtained through the Jacobians of the two robots and a practical rule of composition has been provided. The results achieved for a simple case study have validated the theoretical conclusions in view of the physical interpretation of the kineto-statics of a dual robot system. It has been conjectured that the proposed definitions can be extended to the multi-robot case, although the formulation of internal forces is

not straightforward. This issue, along with the analysis of different types of cooperation (e.g. loose, soft) and the formulation of dynamic manipulability ellipsoids, will constitute the subject of further investigation.

References

[1] J. Y. S. Luh and Y. F. Zheng, "Constrained relations between two coordinated industrial robots for motion control," *Int. J. Robotics Research*, Vol. 6, No. 3, pp. 60–70, 1987.

[2] P. Dauchez and M. Uchiyama, "Kinematic formulation for two force-controlled cooperating robots," *3rd Int. Conf. on Advanced Robotics*, Versailles, France, Oct. 1987.

[3] M. Uchiyama, N. Iwasawa, and K. Hakomori, "Hybrid position/force control for coordination of a two-arm robot," *4th IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, Mar.–Apr. 1987.

[4] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," *1st Int. Symp. on Robotics Research*, Eds. M. Brady & R. P. Paul, MIT Press, Cambridge, MA, pp. 735–748, 1984.

[5] S. Lee and A. K. Bejczy, "Dual redundant arm system manipulability," *NATO Advanced Research Workshop on Robots with Redundancy: Design, Sensing and Control*, Salò, Italy, June–July 1988.

[6] Y. Nakamura, K. Nagai, and T. Yoshikawa, "Mechanics of coordinative manipulation by multiple robotic mechanisms," *4th IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, Mar.–Apr. 1987.

[7] S. Chiu, "Control of redundant manipulators for task compatibility," *4th IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, Mar.–Apr. 1987.

[8] L. Sciavicco and B. Siciliano, "Solving the inverse kinematic problem for robotic manipulators," *6th CISM-IFToMM Ro.Man.Sy*, Cracow, Poland, Sept. 1986.

Appendix A

The matrix $J_a^{T\dagger}$ defined in (24), by substituting the expressions of W in (5) and J_{12} in (21), becomes

$$J_a^{T\dagger} = \begin{pmatrix} I & 0 & I & 0 \\ -R_{1a} & I & -R_{1a} & I \end{pmatrix} \begin{pmatrix} J_1^{T\dagger} & 0 \\ 0 & J_2^{T\dagger} \end{pmatrix}. \tag{A-1}$$

The matrices $J_i^{T\dagger}$ can be partitioned by rows as

$$J_i^{T\dagger} = \begin{pmatrix} (J_i^{T\dagger})_f \\ (J_i^{T\dagger})_n \end{pmatrix} \qquad i = 1, 2 \tag{A-2}$$

where the subscripts f and n refer to forces and moments respectively. Plugging (A–2) in (A–1) gives

$$J_a^{T\dagger} = \begin{pmatrix} J_1^{T\dagger} & J_2^{T\dagger} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -R_{1a}(J_1^{T\dagger})_f & -R_{2a}(J_2^{T\dagger})_f \end{pmatrix} \tag{A-3}$$

which, by virtue of (31), can be compactly written as

$$J_a^{T\dagger} = \begin{pmatrix} \tilde{J}_1^{T\dagger} & \tilde{J}_2^{T\dagger} \end{pmatrix}. \tag{A-4}$$

The matrix $J_a J_a^T$ in (26) can then be computed. First, one obtains

$$J_a^T = \begin{pmatrix} J_1^\dagger (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-1} \\ J_2^\dagger (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-1} \end{pmatrix} \tag{A-5}$$

where the "~"'s have been dropped without loss of generality. Eq. (A–5) leads to

$$J_a J_a^T = (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-T} J_1^{T\dagger} J_1^\dagger (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-1} + (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-T} J_2^{T\dagger} J_2^\dagger (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-1} \quad \text{(A–6)}$$

that can be compacted into

$$J_a J_a^T = (J_1^{T\dagger} J_1^\dagger + J_2^{T\dagger} J_2^\dagger)^{-T}. \quad \text{(A–7)}$$

By virtue of the property $J_i^{T\dagger} J_i^\dagger = (J_i J_i^T)^{-1}$, eq. (A–7) directly leads to (30).

Appendix B

The matrix J_r^T defined in (33), by substituting the expression of J_{12} in (21) and V in (7), becomes

$$J_r^T = \begin{pmatrix} J_1^T & 0 \\ 0 & J_2^T \end{pmatrix} \begin{pmatrix} I & 0 \\ R_{1a} & I \\ -I & 0 \\ -R_{2a} & -I \end{pmatrix}. \quad \text{(B–1)}$$

The matrices J_i^T can be partitioned by columns as

$$J_i^T = ((J_i^T)_f \quad (J_i^T)_n) \qquad i = 1, 2. \quad \text{(B–2)}$$

Plugging (B–2) in (B–1) gives

$$J_r^T = \begin{pmatrix} J_1^T \\ -J_2^T \end{pmatrix} + \begin{pmatrix} 0 & (J_1^T)_n R_{1a} \\ 0 & -(J_2^T)_n R_{2a} \end{pmatrix} \quad \text{(B–3)}$$

which, by virtue of (38), can be compactly written as

$$J_r^T = \begin{pmatrix} \hat{J}_1^T \\ -\hat{J}_2^T \end{pmatrix}. \quad \text{(B–4)}$$

Computing the matrix $J_r J_r^T$ in (35) directly leads to (37).

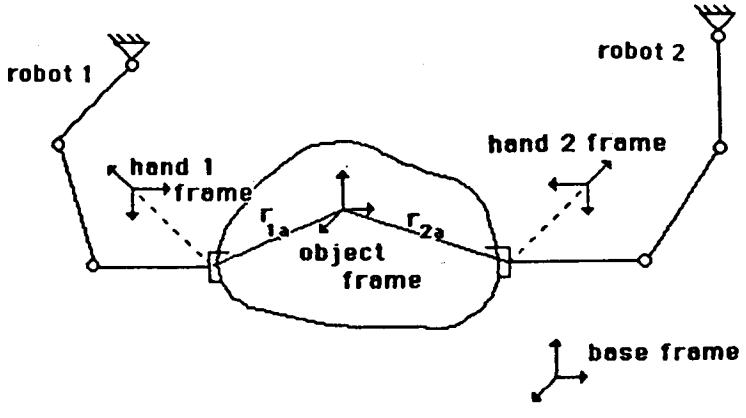


Fig. 1 – A dual cooperative robot system

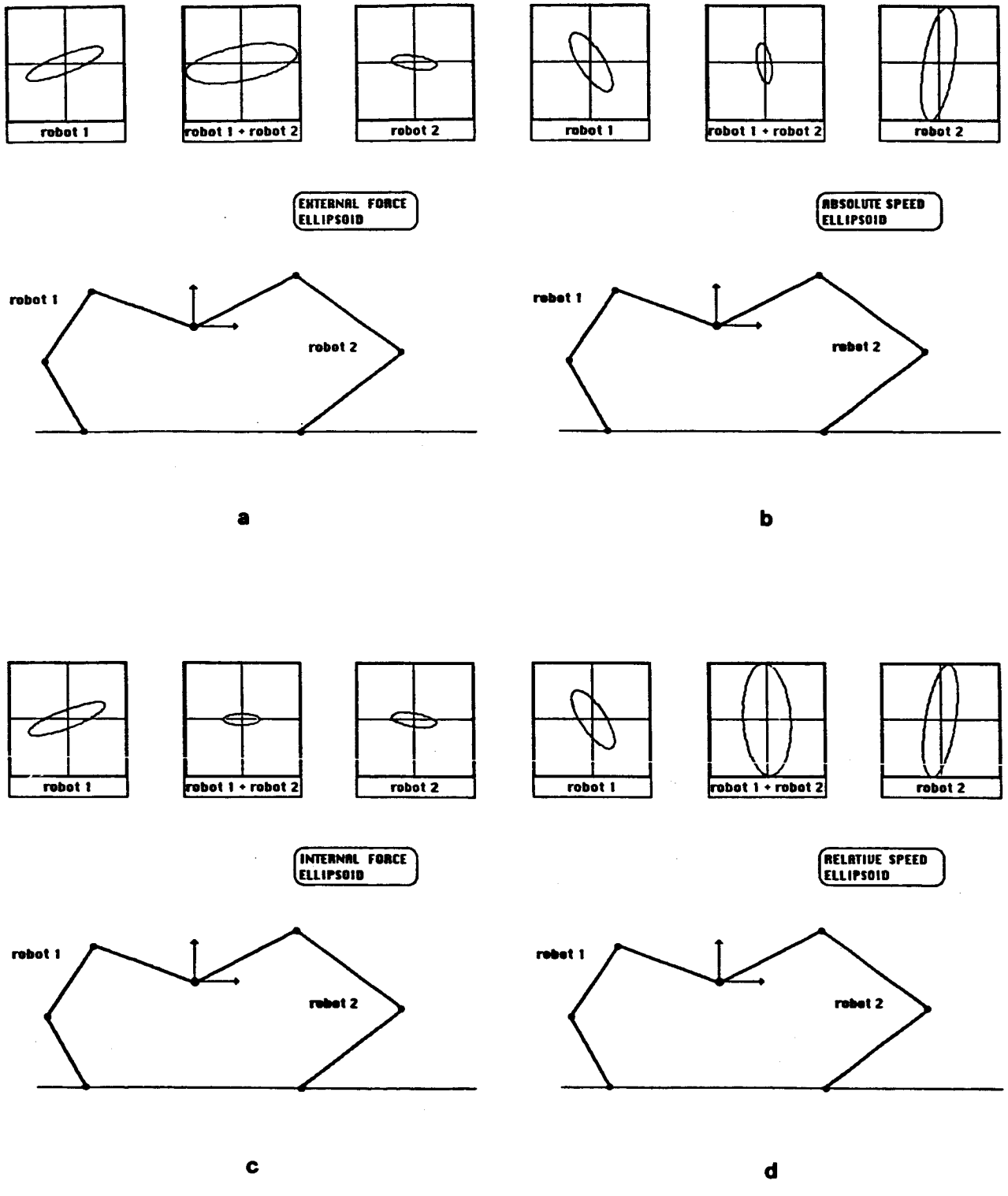


Fig. 2 – Manipulability ellipsoids for a first configuration

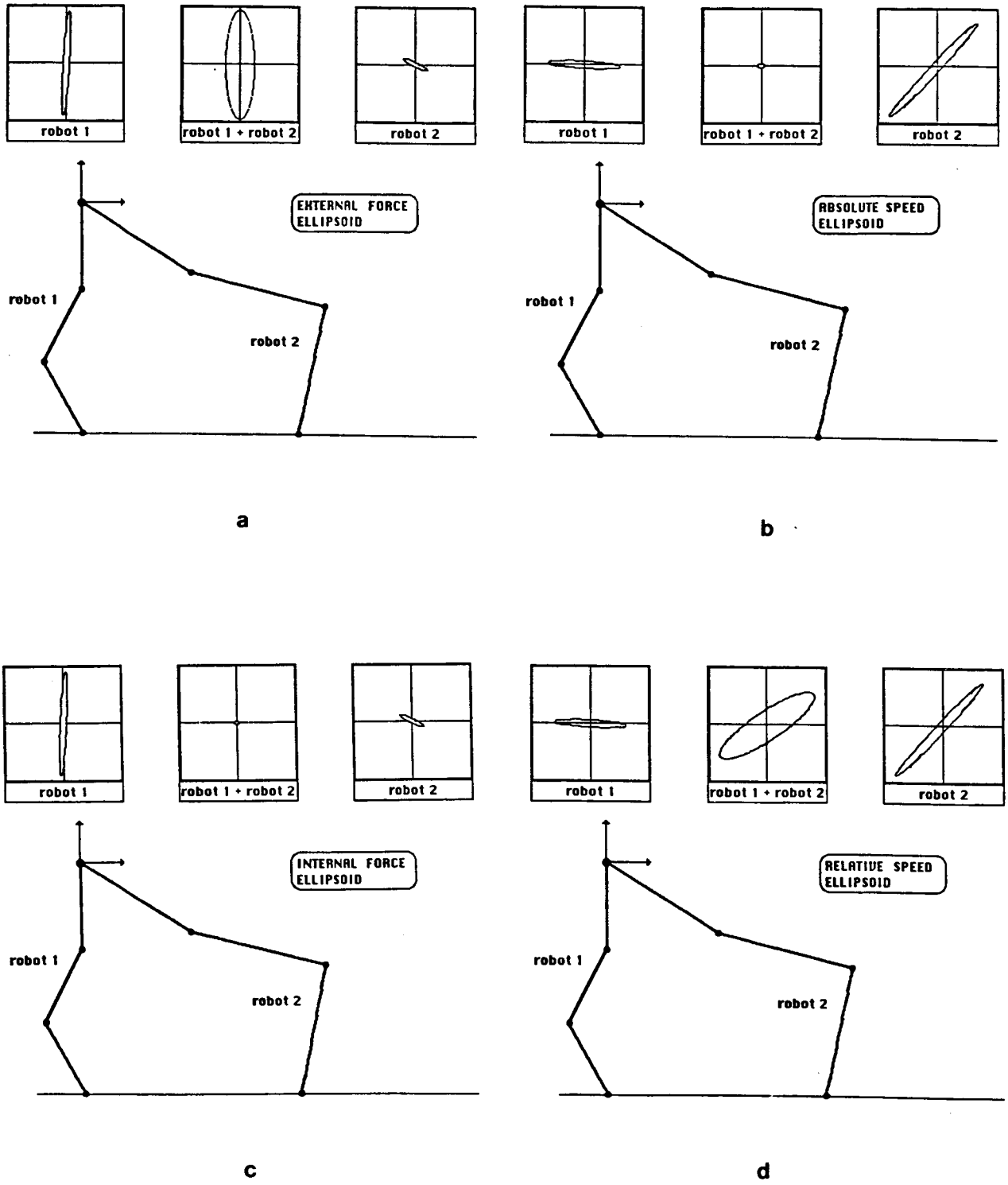


Fig. 3 – Manipulability ellipsoids for a second configuration

Controlling Multiple Manipulators Using RIPS

Yulun Wang Steve Jordan Amante Mangaser Steve Butner

Center for Robotic Systems in Microelectronics
University of California, Santa Barbara

Abstract

A prototype of the RIPS architecture - Robotic Instruction Processing System - has been developed at the Center for Robotic Systems in Microelectronics, University of California at Santa Barbara. A two-arm robot control experiment is underway in order to characterize the architecture as well as research multi-arm control. This experiment uses two manipulators to cooperatively position an object. The location of the object is specified by the host computer's mouse. Consequently, real-time kinematics and dynamics are necessary. The RIPS architecture is specialized so that it can satisfy these real-time constraints.

This paper discusses the two-arm experimental set-up. A major part of this work is the continued development of a good programming environment for RIPS. We are working with the C++ language and have favorable results in the targetting of this language to the RIPS hardware.

1. Introduction

RIPS, for Robotic Instruction Processing System, is a specialized computer targetted to meet the real-time computational requirements for advanced robot control [1-3]. Although the architecture assumes no manipulator characteristic or control strategies, its unique structure can extract most of the parallelism inherent to robot control problems.

A prototype system has been designed and built. This system is being used by researchers at the Center for Robotic Systems in Microelectronics at the University of California, Santa Barbara, to experiment with robot control algorithms which were previously too computationally intensive for real-time evaluation.

This paper describes our current efforts in a two-arm cooperative manipulation experiment. This experiment will enable us to analyze the performance of RIPS as well as study two-arm control. Since RIPS is a custom system, a major part of this work has been the development of a programming environment which can support robotic research. Section 2 gives a brief description of the RIPS architecture. Section 3 describes the two-arm experiment. Section 4 discusses the programming environment which we are developing, and section 5 offers some concluding remarks.

2. The RIPS Architecture

RIPS is a multiprocessor architecture where a tightly coupled cluster of processors is allocated to each dynamically coupled system. Multiple clusters are used to control multiple manipulators. The system level architecture is shown in Figure 1.

Within a cluster, a private bus controlled by a custom I/O (DMA) handler provides high-speed interprocessor communication. Data transfers between processors of the same cluster require a 3 microsecond set-up time, and 400 nanoseconds per 32-bit word transfer. Therefore, multiple transfers are possible within the servoing update cycle.

Communication between clusters is supported by a standard asynchronous bus (VME), which operates at a slower speed. This bus is used for higher level, and hence slower, communication. For example, this bus can be used to coordinate the motion of multiple mechanisms.

The RIPS architecture uses extensive parallel processing to increase the real-time execution speed of robot control algorithms. Before parallel processing can be applied, however, it is important to realize that parallelism can be exploited at many different levels. For example, partitioning the target problem into multiple sub-problems, and simultaneously executing the sub-problems is considered *job level* parallelism. Whereas, pipelining the instruction execution of a processor is considered *intra-instruction level* parallelism. A detailed understanding of the target problem is essential before parallel processing can be effectively applied.

In the RIPS system, each of the subsystems simultaneously executes a different part of the robot control problem. The general schema is a convenient user interface at the host level, which issues trajectory-level commands to the robotic processor(s). The robotic processor evaluates the inverse kinematic and inverse dynamic equations for a particular manipulator, and uses its I/O handler for interprocessor communication. The servo controllers run independently using (optionally) a higher speed update cycle to servo about trajectory set points.

The robotic processor is a custom processor designed to exploit parallelism in robot kinematics and dynamics. After examining kinematic and dynamic equations, we found that they can be efficiently formulated using three-dimensional vector equations. In fact, any rigid body dynamics problem can be expressed in three-dimensional vector notation. An intuitive reason for this intrinsic structure is that these equations explain the motion of three-dimensional bodies in a three-dimensional space. Consequently quantities like positions, velocities, accelerations, forces, and moments are most conveniently described by 3-D vectors. A detailed explanation of the robotic processor's operation can be found in [3].

A prototype RIPS system has been built and is currently being used for a two-arm manipulation experiment. The current configuration consists of a SUN 3/140 host computer, one robotic processor, one I/O handler, and one servo controller.

3. A Two-Arm Experiment

Advances in multi-arm cooperative manipulation will enhance the capabilitties of robots tremendously. One of the major difficulties which prevents the development of such systems is that the computational requirements quickly become overwhelming, even with moderately complex manipulators performing seemingly simple tasks. We felt that RIPS can offer good experimental data to this area of research.

Our experiment is designed both to provide insight into two-arm control as well as to test the RIPS architecture. The two-arm set-up is shown in Figure 2. Each arm is a five-bar-link direct-drive mechanism, which was designed and built at the CRSM [4]. Three direct-drive motors are used to control each manipulator. Even though these manipulators can move only in a horizontal plane, our control programs assume general 3-D capabilities. This is so that our results can be extrapolated and applied to more complex mechanisms.

Our experiment demonstrates real-time two arm cooperation. The SUN host's mouse generates a trajectory for an object which is held by both arms. Since the motion of the mouse is not preplanned, the inverse kinematics must be computed in real time. If high speed manipulation is required, real-time inverse dynamics is also needed.

3.1 Control Strategy

When two arms cooperatively manipulate an object, the entire system (i.e. the two arms and the object) becomes dynamically coupled. If the inverse dynamics of this system is solved with a single recursive algorithm, the computational burden becomes enormous. Nakamura [5] proposed a dynamic force control method which allows parallel processing to be used. A block diagram of this method controlling two cooperating manipulators is shown in Figure 3. At the beginning of each update cycle, the simple dynamics of the object is calculated to determine the trajectory of each manipulator. This information is fed to the servo blocks. The servoing of each arm, which includes the inverse dynamics and kinematics, is computed in parallel, reducing the computation time considerably.

Our prototype system has only one copy of each processor. As discussed earlier, ideally one cluster of processors is assigned to each manipulator. However, due to limited resources we perform the computations for both manipulators with one set of processors. The high computational capacity of these subsystems allows us to maintain a good update rate (~1 ms) anyways.

3.2 Sensor data acquisition and Filtering

The manipulators' actuators and encoders are accessed though a custom interface card. This card has 6 optical encoder input channels and 6 analog output channels. Multiple cards can be used simultaneously if additional joints are to be controlled.

Accurate position and velocity measurements are mandatory for good trajectory control. The interface card uses standard optical encoder feedback to measure joint position. Joint velocity is measured by using a high frequency clock to compute the time between successive rising edges from one of the two optical encoder channels. Velocity is derived by the servo controller with a simple inversion algorithm. Only one of the two channels is used since optical encoders do not always maintain their two channels 90 degrees out of phase. The direction of rotation is also incorporated into the velocity signal by noting the change in position. A 2 pole low-pass filter was necessary to eliminate high-frequency noise.

Figure 4 shows an example of the resulting velocity signal before and after filtering. The corresponding position is displayed for reference. This graph shows a manipulator joint's response to an input step. Note that we underdamped the system to create a more demonstrative picture.

4. Programming Environment

We have spent a good deal of time developing a programming environment for RIPS. A software support will allow RIPS to be used more effectively in robot control research, such as our two-arm experiment. Since our processing subsystems are custom designed, we have had to develop our software starting from a very low level. Fortunately the host operating system is UNIX, so all of the UNIX programming tools are available. These tools have assisted us tremendously in our development efforts.

A UNIX device driver lets us communicate with the different processing subsystems. A menu-driven monitor program assists the user in issuing commands. Figure 5 gives a snapshot of its format. The left column contains the system level commands. Some of these commands have sub-menus which are displayed in the right column when requested. Figure 5 shows the sub-menu for the robotic processor in the right column. This menu is displayed when the user types the command 6 (for rpmenu). Our eventual goal is to develop an interface which appears identical to the UNIX shell. Our shell, however, will understand the additional hardware resources (i.e. robotic processors, I/O handlers, and servo controllers) and know how to use them. With this, a user can run a program which is automatically executed on the custom hardware.

Good program development support is essential for an experimental system. This begins from the lowest-level translators. A TMS320C25 assembler donated from Texas Instruments was ported from MS-DOS to UNIX so that we can write I/O handler and servo controller programs. We had to write a custom assembler for the robotic processor because of its unique instruction set. This assembler was designed so that the instruction set can be easily modifed. This has turned out to be a very useful; we have already updated the robotic processor's instruction set twice.

4.1 High-Level Language Programming

High-level language support allows other users to easily write programs for the system. We have already developed a number of large assembly language programs and are well aware of the inconveniences of assembly language programming.

We are first targetting a high-level language to the robotic processor since it executes the most difficult algorithms. The I/O handler and servo controller programs should remain fairly simple, hence assembly language programming is not too cumbersome. The I/O handler only executes a fairly simple polling program for interprocessor communication. The servo controller interfaces to the manipulators, and calculates simple control laws. Eventually we will provide high-level support for these subsystems.

We have decided to support C++ as the high-level language [6]. The main advantages of C++ are that it is object-oriented, it supports user-defined types, and it permits function overloading. Instead of starting from scratch, we are retargetting the GNU compiler from the Free Software Foundation [7]. This compiler is distributed freely in source code form, and has already developed a reputation for its high quality code. The GNU optimizing C and C++ compilers share the same retargettable back end, but have different front ends.

The robotic processor has an original architecture which creates some original compiler problems. For example, the same register file is used to store both vector and scalar operands. Therefore, the compiler must be taught the relationship between vector and scalar registers.

We have already successfully retargetted the GNU C compiler to the robotic processor. The major effort involved writing a machine description which explains the processor's architecture to the compiler. This compiler allows us to write C programs which execute on the robotic processor. However, it cannot exploit the processor's vector capabilities. This limitation is partly due to the C language itself. The advantages of C++ should be helpful in remedying this problem. C++ supports the concept of user-defined types, which is explained in the next section.

4.2 C++, the Robotic Processor, and Robot Control

The robotic processor is optimized for operating on 3-dimensional vectors. Since C++ supports user defined types we can exploit this characteristic by defining a new type, VEC3. VEC3 stands for 3-dimensional vector. This new type enables the compiler to recognize the three-fold parallelism inherent in the program, and exploit the underlying hardware. This is best explained with an example.

Computing the Jacobian matrix

The Jacobian matrix relates joint velocities to Cartesian velocities, and is used throughout various robot control techniques. Orin and Schrader developed a very efficient algorithm for computing the Jacobian [8]. The equations which implement the algorithm are:

$$^{N+1}\mathbf{U}_{N+1} = \mathbf{I}$$

$$^{N+1}\mathbf{U}_{i-1} = {}^{N+1}\mathbf{U}_i \, {}^{i-1}\mathbf{U}^T_i \qquad i = (N+1), \ldots 2, 1$$

$$^{N+1}\boldsymbol{\gamma}_{i-1} = {}^{N+1}\mathbf{U}_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad i = 1, 2, \ldots N \quad \text{revolute joint}$$

$$^{N+1}\boldsymbol{\gamma}_{i-1} = 0 \qquad i = 1, 2, \ldots N \quad \text{prismatic joint}$$

$$^{N+1}\mathbf{r}_{N+1} = 0$$

$$^{N+1}\mathbf{r}_{i-1} = {}^{N+1}\mathbf{r}_i - {}^{N+1}\mathbf{U}_i \, {}^{i}\mathbf{p}^*_i$$

$$^{N+1}\boldsymbol{\beta}_i = {}^{N+1}\boldsymbol{\gamma}_i \times (- {}^{N+1}\mathbf{r}_i) \quad i = 1, 2, \ldots N \quad \text{revolute joint}$$

$$^{N+1}\boldsymbol{\beta}_{i-1} = {}^{N+1}\mathbf{U}_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad i = 1, 2, \ldots N \quad \text{prismatic joint}$$

where the Us are 3-by-3 rotational matrices, the Is are 3-by-3 identity matrices, and the βs, γs, **r**s and **p***s are 3-dimensional vectors. β_i and γ_i form the ith column of the Jacobian. This algorithm starts at the manipulator's end-effector and recursively propagates velocities back towards the base. The resulting Jacobian transforms joint velocities to Cartesian velocities with reference to the manipulator's end effector. Since the Jacobian matrix transforms joint velocities to linear and angular velocities, the algorithm is most efficiently expressed using only 3-D vectors and 3-by-3 rotational matrices.

As mentioned earlier, C++ lets the user define new data types - or *classes* - as well as the operator functions when operating on data of the new types - or *overloading*. The VEC3 data type is instrumental in programming this algorithm so that the robotic processor can exploit the algorithm's inherent parallelism. The three components of a VEC3 object are i, j, and k. These components can be accessed individually as well as collectively. We overloaded the arithmetic operators so that the C++ compiler understands how to manipulate VEC3 objects. Addition, subtraction, multiplication, and division are performed component-wise. In other words, for

addition, the i components are added together, the j components are added together, and the k components are added together. We also overloaded useful combinations of VEC3 objects and scalars. For example, the addition of a VEC3 object and a scalar is defined such that the scalar is added to each component of the vector. Some combinations are meaningless. For example, dividing a scalar by a vector.

Another class called ROT was created to support rotational transformations. A ROT is a 3-by-3 matrix made of three 3-dimensional column vectors, each VEC3s. Each column can be individually accessed as vectors n (normal), o (orientation), and a (approach), which is consistent with standard homogeneous transformation terminology [9]. Furthermore, each component of a vector can be accessed individually. For example, if a ROT ttt was declared, ttt.n refers to column n of the matrix, and ttt.n.i refers to the ith component of the nth column.This gives the programmer a clean way of accessing the different pieces of a matrix.

Having defined these two classes we can write a program which implements the Jacobian algorithm mentioned above. This program generates the Jacobian matrix for a PUMA 560. However, it is easy to modify it for any robot manipulator by changing the value of the constant REVOLUTE.

```
#include <stream.h>
#include <math.h>
#include "3D_classes.h"

const REVOLUTE = 0x7e     // bit code for revolute/prismatic combination for
                          // the PUMA manipulator
main()
{
int i;
float theta[7];

/* Declaration of 6-by-6 Jacobian Matrix */

VEC3 gamma[7];            // gamma[0] is not used
VEC3 beta[7];             // beta[0] is not used

for(i = 1; i < 7; i++){
        gamma[i] = VEC3(0.0, 0.0, 0.0);     // clear Jacobian
        beta[i] = VEC3(0.0, 0.0, 0.0);
}

ROT U[8];                 // initialize rotational matrices with the
U[1] = ROT(PI/2, -PI/2);  // Denivit-Hartenberg joint angle and twist angle
U[2] = ROT(PI/8, 0.0);
U[3] = ROT(PI/7, PI/2);
U[4] = ROT(PI/10, PI/2);
U[5] = ROT(PI/30, PI/2);
U[6] = ROT(PI/20, 0.0);
U[7] = ROT(0.0, 0.0);

VEC3 p[7];                          //initialize translation vectors
p[1] = VEC3(0.0, 0.0, 0.0);         // these vectors point from origin of one
p[2] = VEC3(431.8, 0.0, 149.9);     // coordinate frame to the next
p[3] = VEC3(-20.32, 0.0, 0.0);
```

```
p[4] = VEC3(0.0, -433.07, 0.0);
p[5] = VEC3(0.0, 0.0, 0.0);
p[6] = VEC3(0.0, 0.0, 56.25);
p[7] = VEC3(0.0, 0.0, 0.0);

ROT N_U[8];                          // initialize base to joint rotational matrices
N_U[7] = ROT(0.0, 0.0);              // initialize end-effector rotational matrix to I

VEC3 r[8];                           // initialize base to joint vector
r[7] = VEC3( 0.0, 0.0, 0.0);         // clear vectors
r[6] = VEC3( 0.0, 0.0, 0.0);

for (i = 6; i >= 0; i--){

        theta[i] = RD_THETA;         // read in new theta value
        U[i].load_theta(theta[i]);   // input new theta into U matrix

        N_U[i] = N_U[i+1] * U[i+1].T();
        r[i] = r[i+1] - N_U[i+1] * p[i+1];
}

for (i = 6; i > 0; i--){

        if((0x1 << i) & REVOLUTE) {
                gamma[i] = N_U[i-1].a;          // gamma[i] <- vector a in N_U[i-1]
                beta[i] = cross(gamma[i], -r[i-1]);     // cross product
        }
        else {
                gamma[i] = VEC3(0.0, 0.0, 0.0);
                beta[i] = N_U[i-1].a;
        }
        cout << beta[i] << gamma[i] << "\n";   // output ith column of the Jacobian
}
}
```

This program demonstrates how easily the Jacobian algorithm is written in C++ with the addition of the VEC3 and ROT classes. Furthermore, because the compiler is made aware of these new data types, the final code can exploit the vector nature of the robotic processor.

The arithmetic operators are overloaded to operate on the new data types. Consequently, arithmetic operators can be used instead of function calls, which simplifies the appearance and readability of the program. A couple of specialized functions are also defined. For example, U[i+1].T refers to the transpose of U[i+1], and U[i].load_theta(theta[i]) inserts the new theta value into the appropriate places of the rotational matrix.

C++ supports a mechanism called constructors [6], which are used to initialize user defined classes. VEC3 objects are initialized with the values of the three vector components. ROT data types are initialized with the Denivit-Hartenberg joint and twist angles [9]. Code in 3D_classes.h describes how to generate the 3-by-3 matrix from this information.

5. Conclusion

We are using the RIPS prototype as a test bed for experimenting with two-arm cooperative control. This experiment provides insight to two-arm control research, as well as tests the RIPS architecture. Much of our work has been the development of a programming environment which gives us the tools to perform our experiments.

We have written a monitor program, with a menu interface, for communicating with our custom hardware. A custom interface card which connnects the RIPS system to the two manipulators has been built. This card generates both high-accuracy position and velocity feedback from optical encoder signals.

High-level language support becomes necessary to implement complex control algorithms. We are currently targetting the GNU C++ optimizing compiler to the robotic processor architecture. Our approach will enable us to write programs in high-level constructs and still generate efficient robotic processor vector code.

Acknowledgements

We gratefully acknowledge the contributions of research assistant Vikram Koka for his assistance in software development. We would also like to thank Texas Instruments, Advanced Micro Devices, and Cypress Semiconductor for their generous donations of key components, and the Free Software Foundation for providing the basis for our retargetable optimizing compilers. This work has been funded by the National Science Foundation under grant number 0842415 and by the Allen-Bradley Division of Rockwell International with matching support of the State of California MICRO program.

References

[1] Butner, S., Wang Y., Mangasar, A., Jordan, S., "Design and Simulation of a Robotic Instruction Processing System," *Proc. of the IEEE Conf. on Robotics and Automation*, Phil., PA., April 1988.

[2] Wang, Yulun and Steven E. Butner, "RIPS: A Platform for Experimental Real-Time Sensory-based Robot Control" to appear in the *Trans. on Sys, Man, and Cybernetics*, 1989.

[3] Wang, Yulun, *RIPS: A Computer Architecture for Advanced Robot Control*, Ph.D. dissertation, University of California, Santa Barbara, May 1988.

[4] Asada, H., and T. Kanada," Design of Direct-Drive Mechanical Arms," *ASME Journal of Vibration, Acoustics, Stress, and Reliability in Design*, Vol 105, No 3., pp 312-316, 1983.

[5] Nakamura, Y., Nagai, K. and Y. Tsuneo, "Dynamic Stability in Coordination of Multiple Robotic Mechanisms," *Int. Journal on Robotics Research*, vol 8., No 2., 1989.

[6] Stroustrup, Bjarne,*The C++ Programming Language*, Addison-Wesley, Reading, Mass., 1986.

[7] Stallman, Richard, *Internals of GNU CC*, Free Software Foundation Inc., 1988.

[8] Orin, David E. and William W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators," *Int. Journal of Robotics Research*, vol. 3, No. 4, Winter 1984.

[9] Paul, R.P., *Robot Manipulators*, MIT Press, Cambridge, Mass., 1982.

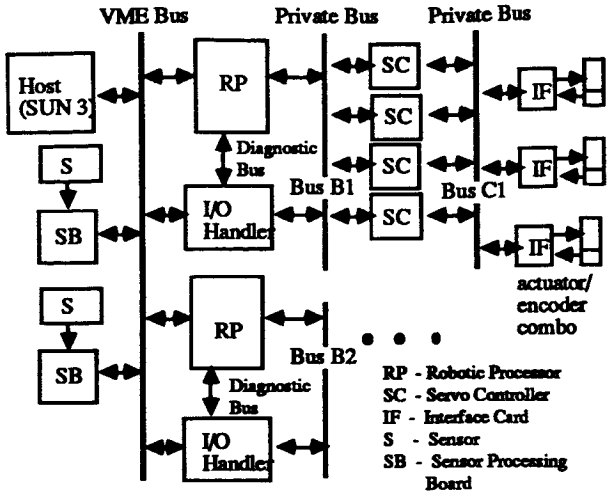


Figure 1. RIPS System Configuration

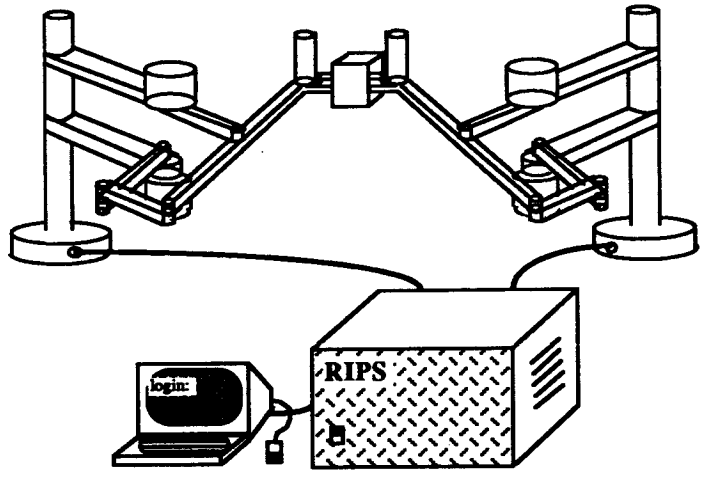


Figure 2. Two-Arm Set up

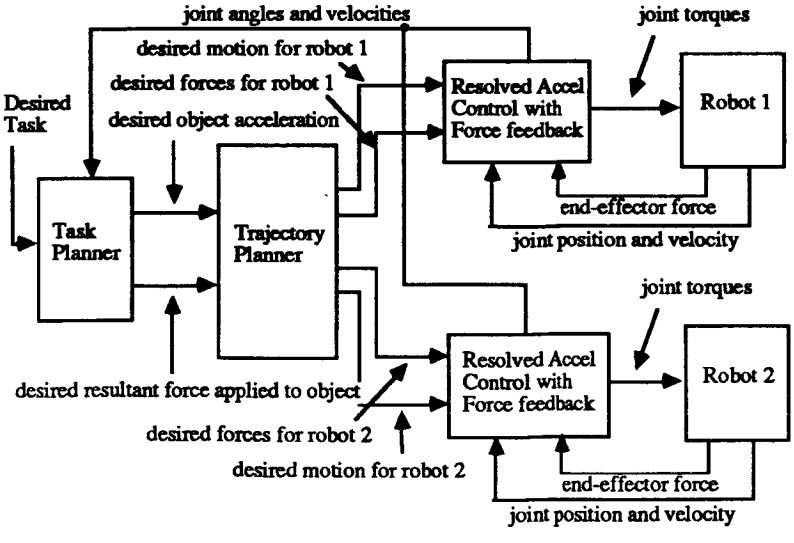


Figure 3. Dynamic Force Control of Two Arms

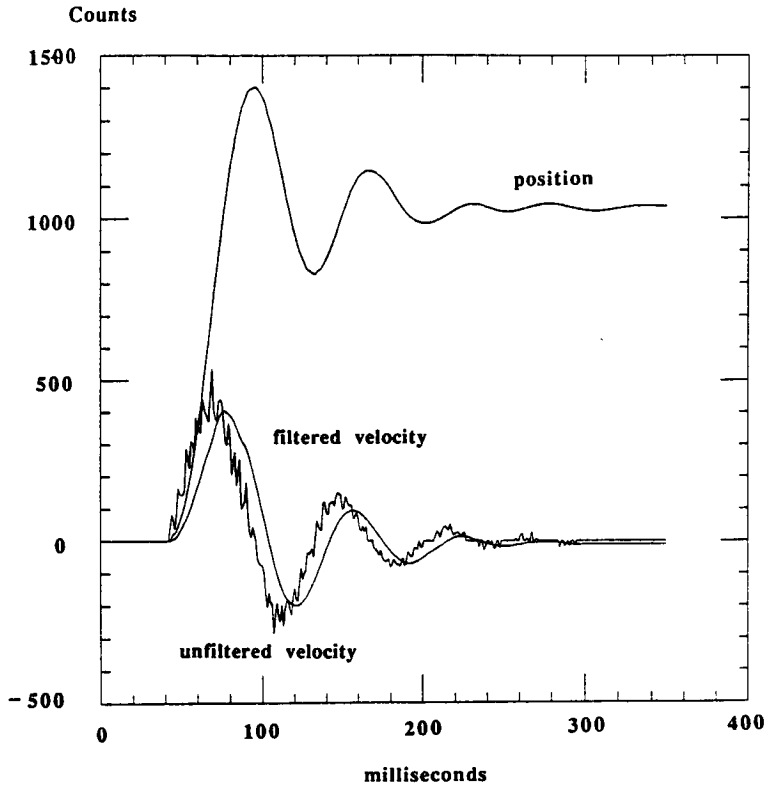


Figure 4. Measured Position and Velocity

```
rips:amante
                    * * * *  RIPS Command Menu  * * * *
RIPS: 1   RP0: 1    IOH: 1   SC: 1-----              Tue Oct 18 18:46:49 1988
=============================================+=============================
                                             |
0: quit:        Terminate session            | 10: rpcurd:   Read RPCU-Host
1: ripsreset:   Reset RIPS                   | 11: rpcurd3:  Read RPCU-Host vectors
2: ripstartup:  Test/start RIPS (NA)         | 12: rpcuwr:   Write RPCU-Host
3: ripstest:    Run Diagnostics (NA)         | 13: rpcuBrd:  Read RPCU-BusB
4: ripsrun:     Start RIPS                   | 14: rpcuBwr:  Write RPCU-BusB
5: ripshalt:    Halt RIPS                    | 15: rppmdown: Download *.lst to RPPM
6: rpmenu:      Issue RP commands            | 16: rppmdump: Dump RP program memory
7: scmenu:      Issue SC commands            | 17: rpdmdown: Download data to RPDM
8: iohmenu:     Issue IOH commands           | 18: rpdmdump: Dump RP data memory
9: testmenu:    RIPS tests                   | 19: rpcrrd:   Read RPCR
                                             | 20: rpdmck:   Check RPDM
                                             | 21: rppmck:   Check RPPM

 Enter command followed by <RETURN>: 19
 SWITCH = 0  RUN/PROG* = 1  HCROR= 0  HCRIR* = 0  HCUIR = 1  HCUOR = 0
 DATA = 0ff
```

Figure 5. Menu-Driven Monitor

TIME OPTIMAL MOVEMENT OF COOPERATING ROBOTS

J. M. McCarthy and J. E. Bobrow

Department of Mechanical Engineering
University of California, Irvine
Irvine, CA 92717

Abstract

This paper examines maximizing the speed of movement, along a prescribed path, of the system formed by a set of robot arms and the object they hold. The actuator torques that maximize the acceleration of the system are shown to be determined by the solution to a standard linear programming problem. The combination of this result with the known control strategy for time optimal movement of a single robot arm yields an algorithm for time optimal movement of multiple robot arms holding the same workpiece.

Introduction

The problem of controlling the movement of an actuated closed kinematic chain is a model for coordinating movement of several arms holding the same object, Luh and Zheng 1985, Tarn, et al. 1987, manipulating an object with the fingers of a mechanical hand, Salisbury and Craig 1982, Li et al. 1988, and controlling the posture of a walking machine, Orin and Oh 1981. In each case an important benefit of the closed chain is the distribution of the load at manipulated body, or workpiece, over the actuators of several different robots.

The problem of specifying the joint torques to achieve a specific movement of the chain is underdetermined, which means a variety of joint torque histories perform the same movement. Recent research focusses on using this freedom to balance the load among the actuators by minimizing the total power consumed by the system along the trajectory, Kreutz and Lokshin 1988, Luh and Zheng 1988 and Zheng and Luh 1988.

In this paper the dynamic indeterminacy that appears in the control of two cooperating 3R robots is resolved by seeking the joint torque history that achieves the least transit time along a specified path. This result generalizes the known solution for the open chain case, Bobrow et al. 1985, and joins minimum time path planning research for closed chains with the existing effort for open chains, Gilbert and Johnson 1985, Shiller and Dubowsky 1988, Bobrow 1988, and Rajan 1985.

Dynamics Equations for Cooperating 3R Robots

In this section the equations of motion for the closed chain formed by a pair of cooperating manipulators are derived, focussing on the case of two planar robots. Previous work in this area includes the studies of the cooperation of walking machine legs Orin and Oh 1981, of dual arm robots Luh and Zheng 1985, and Tarn et al. 1987, Kreutz and Lokshin 1988, and of mechanical hands, Nakamura et al. 1986 and Li et al. 1988.

In this derivation we follow Kreutz and Lokshin 1988 and assume that the end effectors of each robot rigidly hold the workpiece. The combination of these end-effectors and the workpiece is viewed as a single moving rigid body. The joints at each end effector provide the forces and moments that move this body. This viewpoint allows us to formulate the dynamics of each arm and the workpiece independently.

Let the two robots be denoted Robot 1 and Robot 2. The joints of Robot 1 are defined by the position vectors o_1, a_1, and b_1, similarly the joints of Robot 2 are o_2, a_2, and b_2, the rotation angles at each of these joints are denoted θ_i, φ_i, and ψ_i, i=1,2, respectively. See Fig. 1. The lengths the first two links of each robot are $K_i=|a_i-o_i|$ and $L_i=|b_i-a_i|$, i=1,2. The moving body is the rigid link connecting the end joints b_1 and b_2, its length is H. The distance between the base joints o_1 and o_2 is G. For convenience assume that each link is uniform so that its center of mass lies halfway between the joints it contains.

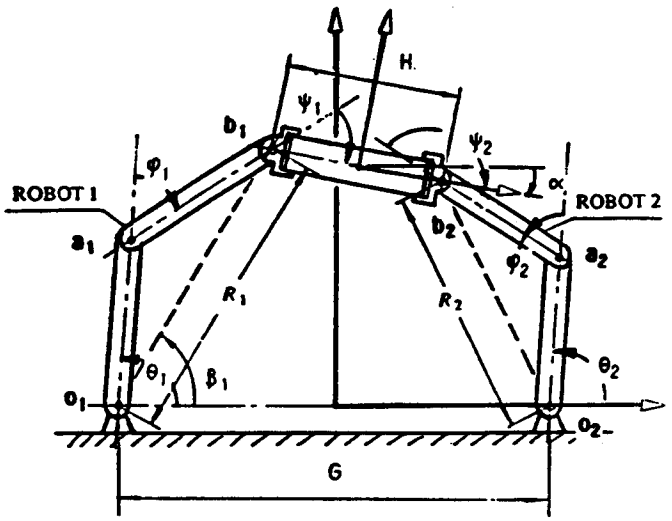


Figure 1. A pair of 3R planar robots holding the same workpiece.

Dynamics of the workpiece. Let $x=(x,y)^T$ be the position of the center of mass of the workpiece and α be its orientation. If $f_1=(X_1,Y_1)^T$ and $f_2=(X_2,Y_2)^T$ are the forces of interaction at the joints b_1 and b_2 and Ψ_1 and Ψ_2 be the motor torques, then the equations of motion of the workpiece are:

$$m\ddot{x} = f_1 + f_2 \ ,$$

$$I\ddot{\alpha} = a_1 \cdot f_1 + a_2 \cdot f_2 + \Psi_1 + \Psi_2 \qquad (1)$$

where

$$a_i = \sigma(H/2\sin\alpha, -H/2\cos\alpha)^T, i = 1,2 \ ,$$

and σ=1 for Robot 1. and σ=-1 for Robot 2. The vectors a_i determine the moment of the joint forces about the center of mass. The dot denotes differentiation with respect to time, and m is the mass of the workpiece while I is its moment of inertia about the center of mass.

Eq. (1) can be written in matrix form by adding α to the coordinate vector x, so that $x = (x,y,\alpha)$ and we obtain

$$[M_0]\ddot{\mathbf{x}} = A_1 f_1 + A_2 f_2 + \tau_0 \ , \tag{2}$$

The torque vector τ_0 is obtained from (1) as

$$\tau_0 = (0,0,\Psi_1 + \Psi_2)^T. \tag{3}$$

and A_1 and A_2 are 3x2 arrays consisting of the 2x2 identity matrix for the first two rows and the vector $\mathbf{a_i}$ in the third row.

The dynamics of each robot. Since the workpiece and end effectors of the robots have been combined into a single body, the equations of motion of each robot reduce to those of a double pendulum with forces applied at the end point $\mathbf{b_i}$. Assemble the joint angles into the vector $\theta_i=(\theta_i, \varphi_i)^T$, and let Θ_i, Φ_i be associated joint torques. The equations of the two robots can be written as

$$[M_i(\theta_i)]\ddot{\theta}_i + \mathbf{h}_i(\theta_i, \dot{\theta}_i) = \tau_i - C_i, \quad i = 1,2 \tag{4}$$

where

$$\tau_i = (\Theta_i - \Psi_i, \ \Phi_i - \Psi_i)^T. \tag{5}$$

$[M_i]$ is the 2x2 mass matrix, $\mathbf{h_i}$ is the vector of Coriolis and gravitational terms, and $\mathbf{C_i}$ arises from the interaction forces at the workpiece. The form of these terms is the same for both robots so we drop the subscript notation:

$$[M_i(\theta)] =$$

$$\begin{bmatrix} I^K + I^L + m^K(K/2)^2 + m^L(L/2)^2 + m^L K^2 + m^L LK\cos\varphi & I^L + m^L(L/2)^2 + \frac{1}{2}m^L LK\cos\varphi \\ I^L + m^L(L/2)^2 + \frac{1}{2}m^L LK\cos\varphi & I^L + m^L(L/2)^2 \end{bmatrix}, \tag{6}$$

$$\mathbf{h}_i(\theta,\dot{\theta}) =$$

$$\begin{pmatrix} -\frac{1}{2}(2\dot{\theta} + \dot{\varphi})\dot{\varphi} m^L LK\sin\varphi - \frac{1}{2}(m^K + 2m^L)gK\cos\theta - m^L g(L/2)\cos(\theta + \varphi) \\ \frac{1}{2}\dot{\theta}^2 m^L LK\sin\varphi - m^L g(L/2)\cos(\theta + \varphi) \end{pmatrix}, \tag{7}$$

where the superscripts K, L refer to the cooresponding link with these lentghs.

The vectors $\mathbf{C_i}$ i=1,2 are the generalized joint torques associated with the forces $\mathbf{f_i}=(X_i,Y_i)$ exerted at $\mathbf{b_i}$, and are given by the equations

$$\mathbf{C_i} = [J_i^T]\ \mathbf{f_i}, \quad i=1,2, \tag{8}$$

where

$$[J_i] = \begin{bmatrix} -K_i s\theta_i - L_i s(\theta_i + \varphi_i) & -L_i s(\theta_i + \varphi_i) \\ K_i c\theta_i + L_i c(\theta_i + \varphi_i) & L_i c(\theta_i + \varphi_i) \end{bmatrix}, \tag{9}$$

is the Jacobian of the i^{th} two link chain. Note s and c denote the sine and cosine functions. The dynamics equations of the workpiece and of the two robots are coupled by the interaction forces at $\mathbf{b_i}$.

The constraint equations. The coordinates, x, y, and α, of the workpiece and the joint angles, θ_i, φ_i, ψ_i, i=1,2, are related by the kinematics equations of the two robots:

$$\begin{aligned} x &= -\sigma G + K_i \cos\theta_i + L_i \cos(\theta_i + \varphi_i) + \sigma(H/2)\cos\alpha, \\ y &= K_i \sin\theta_i + L_i \sin(\theta_i + \varphi_i) + \sigma(H/2)\sin\alpha, \quad i = 1,2 \end{aligned} \tag{10}$$

where σ=1 for Robot 1 and σ=-1 for Robot 2. Given values for $\mathbf{x}$=(x, y, α), (10) can be solved to determine each of the coordinate vectors θ_i=(θ_i, φ_i), i=1,2.

We now determine the relation between the joint velocities $\dot{\theta}_i = (\dot{\theta}_i, \dot{\varphi}_i)$ and accelerations $\ddot{\theta}_i$, and the velocity and acceleration, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$, of the workpiece. This conveniently done by introducing the 2 dimensional vector

$$\tilde{\mathbf{x}} = (x + \sigma G - \sigma(H/2)\cos\alpha, Y - \sigma(H/2)\sin\alpha)^T \tag{11}$$

The derivative of (10) can now be written as

$$\dot{\tilde{\mathbf{x}}} = [J_i]\dot{\theta}_i, \quad i = 1,2 \tag{12}$$

and

$$\ddot{\tilde{\mathbf{x}}} = [\dot{J}_i]\dot{\theta}_i + [J_i]\ddot{\theta}_i, \qquad i = 1,2 \tag{13}$$

where $[J_i]$ is the Jacobian of each robot, Eq. (9), and $[\dot{J}_i]$ is its derivative with respect to time. For a given position, velocity and acceleration, $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$ of the workpiece, we can compute $\tilde{\mathbf{x}}$, $\dot{\tilde{\mathbf{x}}}$, $\ddot{\tilde{\mathbf{x}}}$, and solve equations (12) and (13) to determine $\dot{\theta}_i$ and $\ddot{\theta}_i$.

Time Optimal Control

The problem of controlling the cooperating robot pair so that the workpiece traverses a specified path in minimum time is a generalization of work presented in Bobrow et al. 1985. Also see Dubowsky and Shiller 1988 and Shin and McKay 1984.

First, we assemble the equations of motion of the workpiece and the two arms into a single set of equations by introducing the seven dimensional vector $\mathbf{q} = (\mathbf{x}, \theta_1, \theta_2)^T$ --note $\mathbf{x}$ has three components and θ_i each have two components. The equations of motion of the system become

$$[M]\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = [B]\tau - [C]\mathbf{f} \tag{14}$$

The 7x7 system mass matrix [M] has $[M_0]$, $[M_1]$ and $[M_2]$ along its diagonal.

The vector $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ is simply $\mathbf{h} = (0, 0, 0, h_1, h_2)^T$ obtained from (7). The system torque vector $\tau = (\Theta_1, \Phi_1, \Psi_1, \Theta_1, \Phi_2, \Psi_2)^T$ is six dimensional and [B] is the 7x6 matrix that maps it to the three torque vectors τ_0, τ_1, τ_2:

$$[B] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{15}$$

The four dimensional vector $\mathbf{f}=(f_1,f_2)^T$ represents the interaction forces at the workpiece. The 7x4 matrix [C] is obtained from Eqs. (2) and (8) as

$$[C] = \begin{bmatrix} -A_1 & -A_2 \\ J_1^T & 0 \\ 0 & J_2^T \end{bmatrix} \tag{16}$$

Note that A_1 and A_2 are 3x2 arrays, while J_1^T and J_2^T are 2x2 arrays.

The next step is to introduce a path parameter s which identifies the position of the workpiece as it moves along the specified trajectory. The system equations of motion (14) are written in terms of this parameter. The goal is to determine the maximum acceleration $\ddot{s}$ along the path that is achievable without exceeding the torque limits of the joint motors.

The path parameter. Since it is assumed a path has been specified, the parameterized vector $\mathbf{x}(s) = (x(s), y(s), \alpha(s))^T$ is a known function of some parameter s. We seek the function s(t) that minimizes the transit time without exceeding the maximum torque attainable at each joint. Since $\mathbf{x}(s)$ is given, we can determine $\tilde{\mathbf{x}}(s)$ from (11), and obtain its derivatives in the form

$$\dot{\bar{x}} = (d\bar{x}/ds)\dot{s}$$
$$\ddot{\bar{x}} = (d^2\bar{x}/ds^2)\dot{s}^2 + (d\bar{x}/ds)\ddot{s} \ . \tag{17}$$

Using Eqs. (12) and (13) we obtain

$$\dot{\theta}_i = [J_i]^{-1}(d\bar{x}/ds)\dot{s}$$
$$\ddot{\theta}_i = [J_i]^{-1}\left\{(d^2\bar{x}/ds^2)\dot{s}^2 + (d\bar{x}/ds)\ddot{s} - [\dot{J}_i][J_i]^{-1}(d\bar{x}/ds)\dot{s}\right\}. \tag{18}$$

Ultimately, we obtain $\ddot{q} = (\ddot{x},\ \ddot{\theta}_1,\ \ddot{\theta}_2)$ in the form

$$\ddot{q} = \ddot{s}\,q_1(s,\ \dot{s}) + q_2(s,\ \dot{s}) \ , \tag{19}$$

where $q_1(s,\ \dot{s})$ is the vector of elements that multiplies $\ddot{s}$ and $q_2(s,\ \dot{s})$ is the vector of remaining elements.

The equations of motion (14) can now be written in terms of the path parameter, s, as:

$$[M]\,q_1\ddot{s} + g(q, \dot{q}) = [B]\,\tau - [C]\,f, \tag{20}$$

where

$$g(q, \dot{q}) = [M]q_2 + h(q, \dot{q}).$$

For given values of s and $\dot{s}$, Eq. (20) becomes a set of seven linear equations in the eleven unknowns $\ddot{s}$, τ, and f.

The linear programming problem. The optimal control problem now reduces to computing the torques $\tau = (\Theta_1,\ \Phi_1,\ \Psi_1,\ \Theta_2,\ \Phi_2,\ \Psi_2)$ that provide the maximum (or minimum) acceleration $\ddot{s}$ for each position and velocity, $(s,\ \dot{s})$, of the workpiece, subject to the constraint that the equations of motion (20) are satisfied. Note that because the cooperating robots form a three degree of freedom system (7 coordinates - 4 constraints), for a given acceleration $\ddot{s}$ the torques are not in general unique.

The problem of maximizing the acceleration can be posed as a standard problem in Linear Programming, Thie 1979. That is, a vector $y = (\ddot{s},\ \tau,\ f)^T$ is sought that maximizes (or minimizes) the scalar acceleration

$$\ddot{s} = by, \qquad b = (1,0,0,0,0,0,0,0,0,0,0) \tag{21}$$

The vector y is subject to the linear constraints due to the equations of motion

$$[A]\,\mathbf{y} = \mathbf{c}, \tag{22}$$

where from (20) we have

$$\begin{aligned} [A] &= \left[[M]\,\mathbf{q}_1, -[B], [C]\right] \\ \mathbf{c} &= -\mathbf{g}. \end{aligned}, \tag{23}$$

Bounds on the magnitudes of the components of y are also part of the standard linear programming problem, which is how torque limits on the motors are introduced. Notice that we can also introduce bounds on the magnitude of the forces acting on the workpiece. The arrays [A], and c are known constants for every position and velocity $(s, \dot{s})$, so a standard Linear Programming algorithm can be used to determine y that maximizes $\ddot{s}$, in (21) subject to the constraints in (22).

The solution of this problem provides the set of joint torques τ as well as the interaction forces $\mathbf{f}_1$ and $\mathbf{f}_2$ that yield extreme values for the acceleration, $\ddot{s}$, of the workpiece, for each point $(s, \dot{s})$ in phase space. The result is the ability to compute the acceleration bounds $f(s, \dot{s})$ and $g(s, \dot{s})$,

$$f(s, \dot{s}) \leq \ddot{s} \leq g(s, \dot{s}), \tag{24}$$

such that the dynamics equations (20) are satisfied.

The control strategy. With the ability to compute the maximum and minimum accelerations attainable by the cooperating robot system, the control strategy established in Bobrow et al. 1985 can be applied to achieve a time optimal movement. The essential idea is to always drive the system at its maximum acceleration or deceleration. From this point of view, the problem reduces to the computation of the points at which the shift from acceleration to deceleration and back again occur along the path. To find these switching points, the acceleration equation,

$$\ddot{s} = g(s, \dot{s}), \tag{25}$$

is integrated forward in time from the initial position, and the deceleration equation,

$$\ddot{s} = f(s, \dot{s}), \tag{26}$$

is integrated backward in time from the final position to find when they intersect in phase space. This intersection determines the value of s along the path at which the controller shifts from acceleration to deceleration. In phase space the relation $f(s, \dot{s}) = g(s, \dot{s})$ defines a curve that represents the maximum velocity attainable by the workpiece. It can happen that this maximum velocity constraint is violated before the intersection of the solutions to (25) and (26) occurs. Bobrow et al. 1985 show how to determine intermediate switching points when this occurs, so that time optimal movement is maintained.

Conclusion

In this paper, the time optimal control problem is formulated for the case of two cooperating planar 3R robots. The control strategy is shown to be a generalization of the time optimal control of a single robot arm. The problem of finding minimum time paths for the cooperating 3R arms is a similar generalization of the problem of finding minimum time trajectories for single robot arms. Given the

ability to measure transit times for various paths between two desired positions of the workpiece held by several robots, a nonlinear optimization routine can vary the trajectory, while avoiding obstacles in configuration space, until the minimum time path is found.

Bibliography

Bobrow, J. E., 1988, "Optimal Robot Path Planning Using the Minimum-Time Criterion," *IEEE J. Robotics and Automation*, 4(4):443-450.

Bobrow, J. E., Bodduluri, R. M. C., and McCarthy, J. M., 1988, "Path Planning for Cooperating Planar Robots," UCI Mech. Eng. Technical Report, METR-88-12, submitted to the *IEEE J. of Robotics and Automation.*

Bobrow, J. E., Dubowky, S., and Gibson, J. S. 1985. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robotics Research* 4(3):3-17.

Bottema, O., and Roth, B. 1979. *Theoretical kinematics.* Amsterdam: North Holland Publishing Co.

Chen, Y., and Desrochers, A. A. 1988(Philadelphia). Time-optimal control of two-degree of freedom robot arms. *Proc. IEEE Robotics and Automation Conf.* pp. 1210-1215.

Dubowsky, S., and Shiller, Z. 1984 (Udine, Italy). Optimal dynamic trajectories for robotic manipulators. *Proc. V CISM-IFToMM Symp. Theory and Practice of Robots and Manipulators.*

Gilbert, E. G., and Johnson, D. W., 1985, "Distance Funtions and Their Applications to Robot Path Planning in the Presence of Obstacles," *IEEE J. Robotics and Automation,* 1(1):21-30.

Hayati, S. A. 1987(Pasadena). Dynamics and control of coordinated multiple manipulators. Proc. *JPL Workshop on Space Telerobotics.*

Johnson, D. W., and Gilbert, E. G., 1985(Ft. Lauderdale) "Minimum-time robot path planning in the presence of obstacles," *Proc. 24th Conf. on Decision and Control,* pp.1748-1753.

Kahn, M. E., and Roth, B. 1971. The near-minimum-time control of open-loop articulated kinematic chains. *ASME J. Dyn. Sys. Meas. Contr.*, pp. 164-171.

Kerr, J., and Roth, B. 1986. Analysis of multifingured hands. *Int. J. Robotics Research,* 4(4):3-17.

Kreutz, K., and Lokshin, A., 1988(Atlanta). Load balancing and closed chain multiple arm control for a system of several arms grasping a commonly held object. *Proc. American Control Conf.* pp 2148-2155.

Li, Z., Hsu, P., and Sastry, S. 1988(Atlanta). Dynamic coordination of a multiple robotic system with point contact. *Proc. American Control Conf.* pp. 505-510.

Luh, J.Y.S., and Lin, C.S. 1981. Optimal path planning for mechanical manipulators. *ASME J. Dyn. Sys., Meas., Contr.* 12:142-151.

Luh, J.Y.S., and Zheng, Y-F. 1985. Computation of input generalized forces for robots with closed kinematic chain mechanisms. *IEEE J. Robotics and Automation,* 6:60-70.

Luh, J. Y. S., and Zheng, Y.F., 1988(Atlanta). Load distribution between two coordinating robots by nonlinear programming. *Proc. American Control Conf.* pp. 479-482.

McClamrock, N. H., and Huang, H-P. 1985(Seattle). Dynamics of a closed chain manipulator. *Proc. American Control Conference,* pp. 50-54.

Meier, E. B., and Bryson, A. E., 1987(Monterey), "An efficient algorithm for the time-optimal control of a two-link manipulator," *Proc. AIAA Conf. on Guidance and Control,* pp. 204-212.

Nakamura, Y., Nagai, K. and Yoshikawa, T., 1986(Raleigh) Mechanics of coordinative manipulation by multiple robotic mechanisms. *Proc. IEEE Robotics and Automation Conf.* pp 991-998.

Niv, M., and Auslander, D. M. 1984(San Diego). Optimal control of a robot with obstacles. *Proc. Ammerican Contr. Conf.*, 1:280-287.

Orin, D. E., and Oh, S-Y. 1981. Control of force distribution in robotic mechanisms contraining closed kinematic chains. *ASME J. Dyn. Sys., Meas., and Contr.* 102:134-141.

Rajan, V. T., 1985(St. Louis), "Minimum time trajectory planning," *Proc. IEEE Robotics and Automation Conf.*, pp. 759-764.

Salisbury, J.K., and Craig, J. J. 1982. Articulated hands: Force control and kinematics issures. *Int. J. Robotics Research*, 1(1):4-17.

Shiller, Z., and Dubowsky, S. 1988(Philadelphia). Global time optimal motions of robotic manipulator in the presence of obstacles. *Proc. IEEE Robotics and Automation Conf.*, pp. 370-375.

Shin, K. G., and McKay, N. C., 1984(San Diego). Open-loop minimum-time control of mechanical manipulators and its application. *Proc. American Control Conf.*, pp. 296-303.

Tarn, T. J., Bejczy, A. K., and Yun, X. 1987. Design of dynamic control of two cooperating robot arms: closed chain formulation. *Proc. of the IEEE Int. Conf. on Robotics and Automation,* Raleigh, NC. pp. 7-11.

Tarn, T. J., Bejczy, A. K., and Li, Z. F. 1988(Atlanta). Dynamic workspace analysis of two cooperating robot arms. *Proc. American Control Conference.* pp. 489-498.

Thie, P. R. 1979. *An Introduction to Linear Programming and Game Theory* John Wiley, New York, NY. 335pp.

Vanderplaats, G. N. 1984. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill, New York, NY.

Weinreb, A., and Bryson, A. E., 1985, "Optimal control of systems with hard control bounds," *IEEE J. Automatic Control,* AC30(11):1135-1138.

Zheng, Y. F., and Luh, J.Y.S. 1988(Philadelphia). Optimal load distribution for two industrial robots handling a single object. *Proc. IEEE Robotics and Automation Conf.*, pp. 344-349.

COUPLING OF SYMBOLIC AND NUMERIC SYSTEMS

Reflections on the Relationship Between Artificial Intelligence and Operations Research

Mark S. Fox

Robotics Institute and Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Historically, part of AI's roots lie in Operations Research. It is the goal of this paper to explore how AI has extended the problem solving paradigm developed in OR. In particular, by examining how scheduling problems are solved using OR and AI, we demonstrate that AI extends OR's model of problem solving through the opportunistic use of knowledge, problem reformulation and learning.

1. Introduction

Artificial Intelligence (AI) has come along way during the last 35 years. Like any "new" decision technology, the publicity that heralds its appearance tends to sweep aside technologies, such as Operations Research (OR), that have come before[1]. Not surprisingly, the mere mention of the phrase Artificial Intelligence is liable to elicit from the Operations Research practitioner, a variety of responses from adulation to incredulity. In my experience, neither responses are appropriate, but are due primarily to a lack of knowledge of the field's basic methods. The goal of this paper is to dispel many of the misconceptions that surround AI technology. By examining both approaches to problem solving, both the AI and OR practitioner should obtain a better appreciation of each other's approaches. The ultimate goal being to merge the approaches.

Both OR and AI investigate problem solving methods. As Simon has noted, they share the same roots [19], but diverged at an early point in their history. A common distinction between AI and OR has been that OR deals with well structured problems and AI deals with ill structured problems. I believe it is time to put this distinction to rest.

Looking back at what has been written on the topic, Simon [18] defines a problem as being *well structured* if the means to solving the problem can be operationalized, in a computational sense. This tended to be a concern in the early days of computation[2]. Another view is that an ill structured problem is one which cannot be completely defined. E.g., objective function, constraints. Rapid prototyping is a means of elicting problem structure. A third view [16] is that a problem is ill structured if there only exists *weak* methods to solve it. Weak methods make weak demands on the task environment.

It is the last view that I believe is important. It primarily focuses on the performance of the problem solver. Weak methods tend to perform poorly because they are unable to reduce search complexity. Weak methods are not unique to AI nor OR. They each have their share of weak and strong methods. The real issue is what is the nature of the strength behind OR and AI problem solving theories?

One popular view is that AI problem solving draws its strength from heuristics. Glover explores the role of heuristics in [11], Grant also extolls the importance of heuristics in constructing an aircraft engineering management system, FIXER [12]. Another view is that AI methods are based on heuristic search and symbolic models vs OR's optimization a la linear programming and

[1] A concern also raised by Sutherland [22].

[2] for example, the issue of effective computability as described in Church's thesis [3].

quantitative modeling [19]. The view that will be explored in this paper is that the AI's strength is related to but more subtle than either of these notions.

The rest of this paper addresses this issue. The next two sections explore the strength of the OR approach and AI approach to problem solving respectively. A vareity of methods are described for each applied to the problem of factory scheduling. Based upon these examples the subsequent section explores the relationshiop of AI to OR.

2. OR Approach To The Scheduling Problem

Operations Research draws its problem solving strength from two inter-related sources: the modeling of problems using algebraic constraints, and the optimization of these models by means of mathematical programming -- linear programming being the core problem solving technique. There are other important areas of OR, such as queuing theory, simulation and network analysis, but it is mathematical programming that represents the bulk of the research.

In this section, we investigate the representation and solution of the one machine, non-preemptive weighted tardiness scheduling problem: Given a set of n jobs, sequence them through a single machine so that the total tardiness of all the jobs is minimized. We examine a number of problem solving techniques from an adequacy and efficiency perspective.

2.1. Modeling

The one machine scheduling problem can be modeled by a set of algebraic constraints. Given n jobs each with processing time P_i, due date D_i, and weight w_i:

n jobs.
P_i: processing time for job i
D_i: due date for job i
w_i: weight of job i

The objective is to minimize the total weigthed tardiness **T** subject to the following set of constraints:

$$min\ \mathbf{C} = \sum_{j=1}^{n} w_j T_j$$

$$s.t. \sum_{i=1}^{n} (\sum_{k=1}^{j} P_i X_{ik}) = C_j \qquad for\ all\ j$$

$$C_j - D_j = T_j - E_j \qquad for\ all\ j$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad for\ all\ j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad for\ all\ i$$

$$X_{ij} \in \{0,1\} \qquad integrality$$

$$All\ Variables \geq 0$$

The core of this model is the variable $\mathbf{X}_{ij}$ which specifies the scheduling solution using a positional notation. That is, variable $\mathbf{X}_{ij}$ is one if the job is initially in position *i* is scheduled finally in position *j*, otherwise it is zero. The constraints restrict the use of **X** so that no job can be in more than one position.

While not necessarily obvious, the representation is powerful none-the-less.

2.2. Linear Programming

The first step in solving this problem is to see whether linear programming can be applied directly. In this case, the variable $\mathbf{X}_{ij}$ is an integer and cannot take on fractional values. Any solution generated using linear programming may assign fractional values to **X** which does not have an interpretation here.

2.3. Branch and Bound

Since **X** is integer, the most appropriate approach is to consider integer programming. The simplest algorithm is to generate a tree of alternative assignments to the variables $\mathbf{X}_{ij}$. Starting with the root node, the first two arcs would assign the variable $\mathbf{X}_{1,1}$ the values 0 and 1 respectively. Each of their binary branches would assign the variable $\mathbf{X}_{1,2}$ the values 0 and 1 respectively, and so on. At each node, constraints are tested to see if the variables generated to that point satisfy them. In particular, a node may assign a job to more than one position in the queue. Nodes that violate constraints are pruned. The full decision tree has $n!$ leaves. Search, even with pruning, is exponential in the size of the problem.

Another approach is to estimate the optimal solution and use it as stopping criteria in the branching search. This is can be accomplished by relaxing the integer constraint. This creates an L.P. which is easy to solve. The solution is a lower bound. If this solution is very close to our currently best feasible solution, then we have verified we are very near optimal, and do not have to continue the search.

Or we may use the L.P. solution as a starting "superoptimal," and try to make small changes restoring it to feasibility. That is, we search for a nearby feasible point. If so, we are done; if not, do more extended search by dual ascent of similar technique.

The branching procedure can be extended to full branch and bound if there was a way to prove that parts of the tree are inferior, then prune these parts away without ever checking. We can use the L.P. solution as the lower bounding procedure, but would have to solve an L.P. at each node in the search tree.

In each case, the size of the search tree still grows exponentially.

2.4. Lagrangian Relaxation

Another approach to solving the scheduling is to employ Lagrangian relaxation.

> "Lagrangian relaxation is based upon the observation that many difficult integer programming problems can be modeled as a relatively easy problem complicated by a set of side constraints. To exploit this observation, we create a Lagrangian problem in which the complicating constraints are replaced with a penalty term in the objective function involving the amount of violation of the constraints and their dual variables." [8]

The idea is that the resulting integer program is supposed to be easier to solve. Since some constraints are missing, we will get a lower bound on the solution for any value of the lagrange multipliers.

$$min\mathbf{C} = \sum_{j=1}^{n} W_j T_j + \ (1-\sum_{i=1}^{n} X_{ij})$$

$$s.t. \sum_{i=1}^{n} (\sum_{k=1}^{j} P_i X_{ik}) = C_j \qquad \textit{for all } j$$

$$C_j - D_j = T_j - E_j \qquad \textit{for all } j$$

$$X_{ij} \in \{0,1\} \qquad \textit{integrality}$$

$$\textit{All Variables} \geq 0$$

Now we vary the multipliers, looking for values that will make the subsumed constraints true, so that the lower bound at that point will in addition be optimal.

For large problems the complexity of the search is prohitibitive.

2.5. Dispatch Simulation

Another approach to solving the scheduling problem is to use dispatch rules as part of a simulation. A rule is a heuristic that prioritizes jobs in the queue for a machine. In the case where there is a single statistic to be optimized, such as tardiness, dispatch rules have been shown to perform well. But when additional statistics are to be optimized, such as work in process, setups, etc., dispatch rule perform poorly [2].

2.6. OR Methodology

As Simon has noted [19], the emphasis of OR is on mathematical models and their optimization. It appears, at least from our examples, that the dominant optimization method for problems that cannot be solved directly using L.P. is to recast them as linear problems by the process of reformulation. Solutions to this reformulation are then used to guide a more general search process. The solution procedure is as follows:

1. Optimization via Constraint Satisfaction: In this case, a set of linear constraints entail a structure whose properties are known and can be taken advantage of when performing search (e.g., simplex).
2. Reformulation: For complex integer problems, simpler reformulations of the problem are solved in order to provide guidance in solving the more complex version.
3. Simulation: Myopic rules are used to make local decisions hopng that one or more macro statistics will be optimized (e.g., SPT minimizes weighted tardiness).

3. AI Approach To The Scheduling Problem

As in OR, Artificial Intelligence has more than one method of problem solving: deduction, constraint satisfaction and heuristic search. But the "key" insight from which AI draws its strength is best articulated in the Physical Symbol System definition of Newell & Simon [17]. A physical symbol system has the necessary and sufficient means for general intelligent action. A physical symbol system is "a machine which produces through time an evolving collection of symbol structures" and is capable of: Designation and Interpretation.

From this, the Heuristic Search Hypothesis is derived: Solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem solving by search -- that is, by generating and progressively modifying symbol structures until it produces a solution structure [17]. The Problem Space operationalizes the concept of a phsycial symbol system. A problem space is composed of

- **States** which are collections of features that define some situation,
- **Operators**, that transform one state into another, and
- An **evaluation function**, that rates each state in the problem space.

Search begins at an **initial** state, and the problem is solved when a path is found from it to a **goal** state.

In this section we explore a more complex version of the factory scheduling problem. In this version:

- There are n jobs.
- Each job has a process graph, each node in the graph represents an operation, and each path through the graph is a possible process plan.
- Each operations requires zero or more resources, including machines, tooling, material and resources.
- Alternatives may be specified for each required resource.
- Jobs arrive at random times into the factory and have varying due dates.

As in the section on OR, we will first look at the representation of the problem, followed by a successive set of models to solve it.

3.1. Modeling

AI Knowledge representation research focuses on the development of ontologies and semantics that:

- span the set of concepts required to solve a problem,
- precisely and unambiguously represent concepts at all levels of granularity,
- provide a single representation that is understood and used by more than one application (other, more efficient representations may be constructed from this representation as required by an application), and
- be easily understood by the people who construct them.

Consider the following paragraph describing a factory activity:

> The milling operation precedes the drilling operation. It is composed of two steps: setup and run. Setup takes one hour and run time is 10 minutes. Two resources are required. A five poind wrench and an operator. The wrench is only required during setup. The operation is peformed in cost center 48.

Figure 3-1 is an example of the representation of the knowledge in that paragraph. It is relational in form. Figure 3-1 divides the knowledge into two types: activity and state. A state description describes a snapshot of the world before an activity is performed. For example "cost center 48 possess a wrench" is a state description. It must be true in order to enable the milling activity to occur. States and activities are linked via causal relations. A state describes what must be true of the world to enable an activity to occur. In addition, activities may be defined at multiple levels of abstraction. Milling is refined into two sub-activities: the setup and running of the machine. Lastly, we must represent time. Setup, in time, occurs before the milling run. Time is not absolute, it is relative. When describing the factory floor, it is atypical to use absolute time periods, instead, activities are described as preceding each other; and hence, once the time of one activity is determined the time of all the other activities related to it can be determined.

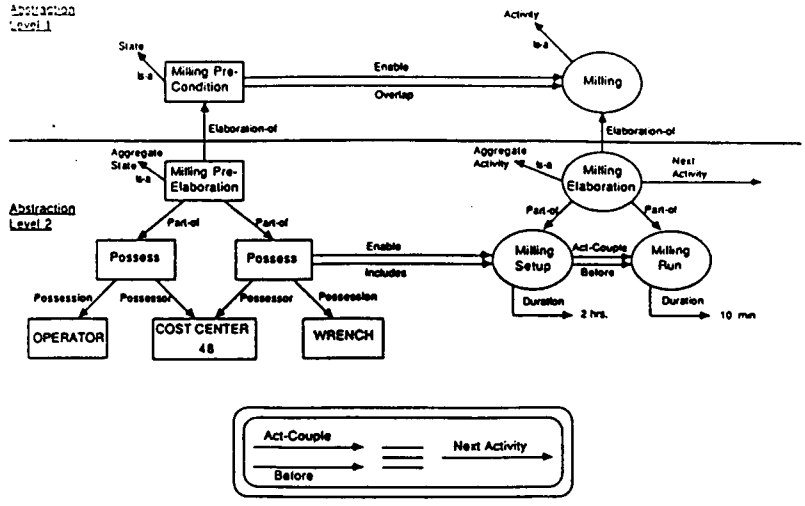


Figure 3-1: Activity Semantic Network

3.2. Constraint Directed Search

Scheduling can be viewed as search through a problem space, where states represent partial schedules, operators extend a partial schedule defined by a state into a new state, and the evaluation function rates each state in the problem space according to the known constraints. Constraint. directed search is a form of search where constraints can be used to specify operators (e.g., operation precedence constraints specify the next operations) and terms of the evaluation function (e.g., a due date constraints measures slack in the schedule). The efficacy of this approach depends on the ability of the constraints to identify the more profitable paths to pursue. Experience has shown that this tends not to be the case.

3.3. Expert Systems

Another important insight in AI, due to Feigenbaum [6], is that the combinatorics of moving through the problem space can be reduced by smarter selection of operators. That is by utilizing knowledge of the domain. Domain knowledge can be represented as elaborations on the conditions of operators, making the application of operators more sensitive to the current context. This leads to the "knowledge-search dichotomy": the more knowledge one has the less search needs to be performed; the less knowledge one has the more search has to be performed to solve the problem.

One source of knowledge is from an expert. For example, rather than generate all possible possible alternative operations everytime scheduling is performed, expertise suggests that we consider alternatives only when capacity becomes scarce on the machines used in the standard plan:

```
IF      Milling is to be scheduled
        AND Capacity is scarce
        AND Queue time is high
        AND Due date is tight

THEN    Consider
                1. Grinding
                2. Subcontracting
```

There are two problems with the expert sytems approach:

1. Problems like factory scheduling tend to be so complex that they are beyond the cognitive capabilities of the human scheduler. Therefore, the schedules produced by the scheduler are poor; nobody wants to emulate their performance.
2. Even if the problem is of relatively low complexity, factory environments change often enough that any expertise built up over time becomes obsolete.

Expert systems appear to be appropriate only when the problem is both small and stable.

3.4. Hierarchical Constraint Directed Search

Since the problem cannot be solved using expert systems, more sophisticated search techniques are required. On approach is to reformulate the problem as a simpler problem whose solution can be used to guide the solution of the original problem. Hierarchical constraint directed search (HCDS) is one method that embodies this approach.

Search is divided into four levels: order selection, capacity analysis, resource analysis, resource assignment. Each level is composed of three phases: a pre-search analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, ISIS uses constraints to bound, guide, and analyze the search.

Level 1 is responsible for selecting the next unscheduled order to be added to the existing shop schedule. Its selection is made according to a prioritization algorithm that considers order type and requested due dates. The selected order is passed to level 2 for scheduling.

Level 2 represents the simpler reformulation of the original problem. It simplifies the problem by removing both resources and constraints from consideration. It performs a dynamic programming analysis of the plant based on current capacity constraints. It determines the earliest start time and latest finish time for each operation of the selected order, as bounded by the order's start and due date. The times generated at this level are codified as operation time bound constraints which serve to influence the search at the next level.

Level 3 solves the original scheduling problem. It selects a particular routing for the order and assigns reservation time bounds to the resources required to produce it. Pre-search analysis begins with an examination of the order's constraints, resulting in the determination of the scheduling direction (either forward from the start date or backward from the due date), the creation of any missing constraints (e.g. due dates, work-in-process), and the selection of the set of search operators which will generate the search space. A beam search is then performed using the selected set of search operators. The search space to be explored is composed of states which represent partial schedules.

Once a set of candidate schedules have been generated, a rule-based post search analysis examines the candidates to determine if one is acceptable (a function of the ratings assigned to the schedules during the search). If no acceptable schedules are found, then diagnosis is performed. Intra-level repair may result in the re-instantiation of the level's search. Pre-analysis is performed again to alter the set of operators and constraints for rescheduling the order. Inter-level repair is initiated if diagnosis determines that the poor solutions were caused by constraint satisfaction decisions made at another level.

Level 3 outputs reservation time bounds for each resource required for the operations in the chosen schedule. Level 4 then establishes actual reservations for the resources required by the selected operations which minimize the work-in-process time.

This approach performs well when there exists adequate capacity in the factory. But in situations where contention for resources was high, the performance of the system was no better than the human scheduler.

3.5. Macro-Opportunistic Constraint Directed Search

In situations where resources are highly contended for, experience has shown that optimizing resource allocation by scheduling operations incident with the resource produces better results than scheduling jobs one at a time. Macro-opportunistic constraint directed search [21, 20] extends HCDS by first analyzing the capacity requirements of all jobs in order to identify whether there is a high degree of resource contention. If contention exists for a resource, then a resource centered scheduler is chosen to schedule the operations incident with it (ususally a dispatch rule simualtion). The job cenered scheduler (i.e., HCDS) is used to scheduler the jobs out from the resource. Opportunism arises out of the systems ability to dynamically determine at any point during the construction of schedules, primary/secondary bottlenecks and take the opportunity to schedule them rather than pursue a strictly job centered approach.

Experiments with this approach has shown that it outperforms both hierarchical constraint directed search and dispatch rule approaches. Secondly, it can efficiently solve typically complex factory scheduling problems.

3.6. Observations

Based on the approaches described above, it is clear that AI problem solving is based upon search and incorporates the following refinements:

- Knowledge of constraints can be used to both guide search (in the form of preferences) and can be used to prune alternatives.
- Reformulation is used to define simpler versions of a problem whose solutions are used to guide search in the more complex case.
- Introspection, that is the systems analysis of its own performance, is used to modify how the system solves the problem in subsequent trials. For example, knowledge of constraint violations can be used to automatically reformulate a problem.
- Opportunism is used to decide where the search is to proceed from. Opportunism can exist at more than one level. At the micro level it can be used to select the next state to extend. At the macro level, it can be used to select the perspective to apply next (e.g., resource vs job).

4. Reflections

It is clear from the earlier sections that both AI and OR are developing strong methods for performing search in the problem space. OR techniques are generally, but not exclusively, constraint directed and quantitatively modeled. AI techniques are generally, but not exclusively, pattern directed and symbolically modeled. Part of the reason for this difference is due to ORs focus on *optimization* and AI on *satisficing*. The former strives to select problems for which it can define high quality, hopefully optimal, and efficient solutions while the latter strives to find high quality and efficient solutions for everyday, messy problems.

It is my view that much of the work in representations and search in AI can be viewed as a natural extension of OR.

Observation 1: AI representations extend OR representations by the processes of abstraction and differentiation.

AI Knowledge representations are qualitative abstractions of underlying quantitative models. Abstractions can be very powerful. They enable the answering of questions that an underlying quantitative model cannot. Lets assume that a quantitative model of time has for each activity to be performed a start time and end time specified. With this model the following question could be answered: "find the start and end time of activity 3 that has the longest duration given start and end times of activity 1 and 2, and knowing that Activity 1 is during activity 2 which is during activity 3.

OBJECTIVE FUNCTION: MAX $et_3 - st_3$

CONSTRAINTS:

$st_1 < et_1$ $st_1 >= st_2$ $st_2 >= st_3$
$st_2 < et_2$ $st_1 <= et_2$ $st_2 <= et_3$
$st_3 < et_3$ $et_1 <= et_2$ $et_2 <= et_3$

WHERE

st_1, et_1, st_2, et_2 are known

ALGORITHM: Simplex

One can view Allen's relational model of time [1] as an abstraction of a quantitative model. In the relational model the temporal relation **during** denotes that one activity is performed during the time that another activity is performed. This relationship can be asserted without knowledge of the actual times of each activity. If we did not know any of the start times or end times for each activity but knew only that Activity 1 is during activity 2 which is during activity 3, we would be able to answer the following question: "Is activity 1 during activity 3?"

GIVEN: A_1, A_2, A_3 are activities, and

A_1 -----d-----> A_2 -----d-----> A_3

where d is the during relation, and
d is transitive

ALGORITHM: Hypothesis Introduction

By its nature, an abstraction is a partial model of an underlying more complete model. In fact, there is a contiuum of partial models one can construct. As demonstrated, each partial model enables the answering of a different set of questions with more or less efficiency.

AI representations also enable the differentiation of concepts that are contained in a quantitative representation. For example, there are many types of temporal relations that can be defined between activities in addition to *during*, for example:

- *before* specifies that the end of one activity occurs before the beginning of another
- *overlap* specifies that the end of one activity occurs after the beginning of anther but before the other ends

Allen [1] identifies 13 types of temporal relations.

Being able to differentiate among concepts contained within but not easily identified in a quantitative model is a requirement for the construction of knowledge based search; the more precise the representation, the more specific the definition of situations for which actions can be defined.

Observation 2: Search is the core method employed in both the AI and OR problem solving models.

By definition, the core problem solving technique of AI is search; it forms the basis of the problem space model. Search plays a similar role in OR. Linear programming is a search technique where vertices in a n dimensional space are visited to find an assignment that optimizes an objective function. Both simplex and karmarkar algorithms exploit knowledge of the structure of the problem space to more quickly find a solution. For integer problems, branch and bound is directly related to AI's AI* algorithm [15]. For non-linear problems, the General Reduced Gradient method is yet another form of search where gradients are used to decide where in the n dimensional space to move to next.

Observation 3: AI extends the model of problem solving by the opportunistic application of situational knowledge of the domain.

The definition of a problem space includes the generation of new states through the application of operators. Conditions of operators are actually descriptions of situations at which a particular decision is to be made. Operators are created as a result of a situational analysis of how to solve a problem.

Operators in AI are opportunistically applied; at each step in the search the operator that best matches the current situation is chosen to extend the search. The transition from one state to another in a problem space in AI programs is rational; at each step the next step is opportunistically made, jumping from one island of certainty to another, and one decision process to another, within the problem space. This is in contrast to how the next decision is selected using OR algorithms. By careful analysis of the strcuture of the problem, OR has devised a more mechanistic approach in deciding what to do next in the search process.

Observation 4: AI extends the model of problem solving from analysis to design.

As Simon has noted [19], OR optimization techniques perform analysis but are unable to perform design. Optimization searches for an assignment of values to variables that optimize an objective function. All variables and their domains are known apriori. But in design new variables and constraints among them are generated during the search process. For example, in the design of a mechanical part, the refinement of the part into subparts introduces new variables (representing the subparts) and constraints among them.

Observation 5: AI extends the model of problem solving to include the automation of problem formulation and re-formulation.

Within OR, problem formulation is the purview of the analyst. AI methods have extended problem solving to include the formulation of problems by the reasoned analysis of hypotheses and the implications. For example, methods of Truth Maintenance [5, 4] enable the representation and management of sets of hypotheses that support deductions. If deductions are found to be unacceptable, hypotheses and their dependent deductions can be withdrawn and alternative hypotheses explored. Consequently, the selection of model parameters are subject to change by the program itself.

In constraint directed search [9, 10], pre and post-analysis of the search space allows for the formulation and re-formulation of the problem via the modification of constraints and the selection of search operators.

Lastly, the SOAR model [14] of problem solving defines the concept of "universal subgoaling" in which new problem spaces are created where re-formulations of the original problem are worked on. The SOAR model supports the opportunistic movement from one problem space to another at each step during problem solving.

Observation 6: AI extends the model of problem solving to include the acquisition of expertise.

L.P. algorithms do not learn from their experience. Running them on the same or similar problem does not result in a net speed up in search. The use of prior experience in increasing a problem solver's performance is never the less important. A variety of methods for learning from prior experience have been used to increase an AI problem solver's performance over time. These include a variety of macro-operations [7, 13] and chunking mechanism [14].

5. Conclusion

Both AI and OR are trying to develop strong methods for performing search in the problem space. OR techniques are generally (but not exclusively) constraint directed and quantitatively modeled. AI techniques are generally (but not exclusively) pattern directed and symbolically modeled. AI methods represent a powerful extension to the OR model of problem solving in two senses. First, the use of situational knowledge in opportunistically guiding search reduces the combinatorics of search in many cases. Secondly, the model of problem solving is extended to include the automation of problem formulation/re-formulation and learning from prior experience.

Acknowledgements

This paper is based on a seminar prepared by Thomas Morton and myself. This work was supported, in part, by the Defense Advanced Research Projects Agency under contract #F30602-88-C-0001, and in part by a grant from Digital Equipment Corporation.

References

[1] Allen, J.F.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.

[2] Baker, K.R.
Introduction to Sequencing and Scheduling.
John Wiley & Sons, 1974.

[3] Brainerd, W.S., and Landweber, L.H.
Theory of Computation.
John Wiley & Sons, 1974.

[4] de Kleer, J.
An Assumption-based TMS.
Artificial Intelligence 28(2):127-162, 1986.

[5] Doyle, J.
A Truth Maintenance System.
Artificial Intelligence :231-272, April, 1979.

[6] Feigenbaum, E.
The Art of Artificial Intelligence.
In *Proceedings of the International Joint Conference on Artificial Intelligence.* Morgan Kaufman, 1977.

[7] Fikes, Hart, and Nilsson.
Learning and Executing Generalized Robot Plans.
Artificial Intelligence 3:251-288, 1972.

[8] Fisher, M.L.
An Applications Oriented Guide to Lagrangian Relaxation.
Interfaces 15(2):10-21, 1985.

[9] Fox, M.S.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
Technical Report, Carnegie-Mellon University, 1983.
CMU-RI-TR-85-7,Intelligent Systems Laboratory,The Robotics Institute, Pittsburgh,PA.

[10] Fox, M.S.
Observations on the Role of Constraints in Problem Solving.
In *Annual Conference of the Canadian Society for the Computational Studies of Intelligence.* University of Quebec Press, 1986.

[11] Glover, F.
Future Paths for Integer Programming and Links to Artificial Intellience.
Technical Report 85-8, Center for Applied Artificial Intelligence, Graduate School of Business, Univesity of Colorado, 1985.

[12] Grant, T.J.
Lessons for O.R. from A.I.: A Scheduling Case Study.
Journal of the Operational Research Society 37(1):41-57, 1986.

[13] Korf, R.E.
Macro-Operators: A Weak Method of Learning.
Artificial Intelligence 26(1):35-77, April, 1985.

[14] Laird, J.E., Newell, A., Rosenbloom, P.S.,
SOAR: An Architecture for General Intelligence.
Artificial Intelligence 33(1):1-64, September, 1987.

[15] Nau, D.S., Kumar, V., and Kanal, L.
General Branch and Bound and Its Relation to A* and AO*.
Artificial Intelligence 23(1):13-28, 1984.

[16] Newell, A.
Heuristic Programming: Ill-Structured Problems.
Progress in Operations Research :360-414, 1969.

[17] Newell, A., and Simon, H.A.
Computer Sciences as Empirical Inquiry: Symbols and Search.
Communications of the ACM 19(3):113-126, 1976.

[18] Simon, H.A.
The Structure of Ill-Structured Problems.
Artificial Intelligence 4:181-200, 1973.

[19] Simon, H.A.
Two Heads Are Better than One: The Collaboration between AI and OR.
Interfaces 17(4):8-15, 1987.

[20] Smith, S.F., and Ow, P.S.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.

[21] Smith, S., Fox, M.S., and Ow, P.S.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4):45-61, Fall, 1986.

[22] Sutherland, J.W.
Assessing the Artificial Intelligence Contribution to Decision Technology.
IEEE Transactions on Systems, Man and Cybernetics SMC-16(1):3-20, 1986.

WHAT KIND OF COMPUTATION IS INTELLIGENCE?
A FRAMEWORK FOR INTEGRATING DIFFERENT KINDS OF EXPERTISE

B. Chandrasekaran

Laboratory for AI Research
Department of Computer & Information Science
The Ohio State University
Columbus, OH 43210

Abstract

The view that the deliberative aspect of intelligent behavior is distinct type of algorithm, in particular a goal-seeking exploratory process using qualitative representations of knowledge and inference, is elaborated. There are other kinds of algorithms that also embody expertise in domains. We discuss the different types of expertise and how they can and should be integrated to give full account of expert behavior.

Extended Summary

1. Intelligence as Computation

The idea that intelligence as a process is algorithmic in character is at the foundation of Artificial Intelligence (AI) as both a science and a technology. With the exception of a new movement called *connectionism*, almost everyone in AI subscribes to this view, especially as intelligence relates to cognitive activities such a problem solving. I have elsewhere[1] discussed connectionism and AI, and noted that connectionist-style systems may have some advantages to offer in modeling perceptual phenomena such as speech or visual recognition, and some cognitive phenomena such as memory retrieval. However, for complex problem solving activities that take place in the *deliberative* mode, the view of the process of intelligence as largely algorithmic, i.e., manipulation of discrete symbol structures, is the most common paradigm. This is mainly because of the fact that "thought", at least as people are aware of it, has a large propositional and hence symbolic content. Discrete symbol structures are generally used to represent the underlying structure of propositions, and hence thoughts. In what follows, I will restrict my remarks to AI and problem solving, in particular expert systems, and I am going to take the algorithmic view for granted for the purposes of this distinction.

Now of course the modern engineer is very familiar with a number of powerful algorithms for various problems. She has cut her teeth on computers, programming

and optimization theories, and has been exposed to linear programming algorithms, finite element analysis, various kinds of numerical simulation programs, etc., etc., each giving precise and useful answers to a number of decision-making problems. Thus a natural question for such a person is whether intelligence is characterized by a special kind of algorithm, different in its power and limitations from the more traditional algorithms she is familiar with. She has probably heard that AI programs are "heuristic" in nature. Additionally, the engineer will surely want to know the extent to which the current AI technology is actually useful in creating intelligent programs. Or is it simply a new programming technology? Even as a programming technology what does it offer in terms of power?

In this talk I plan to offer a view of what characterizes intelligence as a problem solving process, what AI, especially the set of ideas that have come to be called, variously, knowledge-based systems or expert systems, is about and how to understand and characterize the relationship of the algorithmic processes of intelligence to the more traditional types of algorithms that engineers are familiar with.

2. Different points of view about AI

AI is not a unified field of scientific activity; there are significant differences within the field about the nature of intelligence, how to approach its study, and about the goals of AI as a science and technology.

1. For some people AI means understanding the principles behind intelligence as we know it in humans and seeing how that can be computationally supported and exploited for technology. For example, theories that the appropriate architecture for modeling human intelligence is a rule-architecture, that human concepts are stored as a linked network of frames, that problem solving comes in a variety of generic strategies, or that much human problem solving is based on storage, retrieval and modification of large number of cases are theories which attempt to explain some aspects of human intelligence. These theories have become the basis of AI programming or AI-type approaches to solving some problems.
2. There is what one might call a pragmatic view that AI means any programming technique or algorithm that helps us solve complex problems by any combination of techniques, but most commonly by use of so-called "heuristic" search techniques. If the heuristic technique is based on some theory in 1 above, then this view is really close to the view of AI in 1. However, often there is no particular concern that the heuristics reproduce human-like performance, but rather that they reduce the complexity of computation in any way. In fact, human beings aren't particularly good problem solvers in a number of problems such as the

traveling salesman problem for which heuristic algorithms have been invented.

3. Yet others go along with the just mentioned view of AI simply complex algorithms, but restrict the types of problems to be considered as those that humans do particularly well, e.g., problems such as diagnosis, planning, design, and theory formation, which are normally considered tasks that humans excel in. However, the connection to humans is often only at the level of choice of the problem. Once a problem such as diagnosis is chosen, that problem is often treated as an abstract problem for which "normative" algorithms based on some notion of rigor are to be found. There is no particular commitment to use methods of or theories about human-like intelligence. This approach results in a theory of the task-- diagnosis, planning, induction, or whatever -- rather than a theory of intelligence as a process. In this view, people often solve the problems more or less approximately, but the normative algorithm is offered as the ideal algorithm for the problem. A good analogy here is multiplication of two numbers. True some people are good at solving some versions of the multiplication problem in their heads, but studying multiplication as an abstract problem and discovering good algorithms for it has turned out to be very effective. An attempt to do for diagnosis or planning what arithmetic has done for multiplication is often the aim in this approach. If this can be done effectively, and systems can be built based on such algorithms, that would be very welcome from a technological perspective, since one would have algorithms for problems of importance and which until now only humans did well.

   Typically, however, the algorithms obtained by an abstract study of the task of this type tend to be computationally intractable in the general case (unlike the multiplication example). Examples of this are several approaches to diagnosis where the problem is elegantly formulated in some formalism such as logic or statistics, but the algorithms proposed for diagnosis based on the formalism are in general very complex, and heuristic approximations need to be made in particular problems. But these approximations need not, and in general do not, match the power or behavior of human intelligence in these problems, since the original formulation was not based on a theory of intelligence.

I will take the position that intelligence is not an arbitrary collection of complex algorithms as in (2) above, nor is a theory of tasks *per se* a theory of intelligence as in (3). Just because people perform multiplications in their head doesn't mean the theory of multiplication is an AI theory. Just changing multiplication to diagnosis doesn't change the logical character of the situation. That is, while it is certainly technologically useful to obtain good algorithms for diagnosis, independent of a theory of intelligence, we have to be careful about characterizing it as AI. Whenever we have a problem for which we can generate a tractable algorithm about whose behavior we can have some confidence, we should of course use it for our applications. However, where AI is

important is in problems where the normative or formal algorithms have one or both of the following properties: they are intractable, i.e, they are of high computational complexity, or they require information in a form that is not generally available in the domain. In these cases, if there is evidence that human experts solve it, then we can look for AI-type approaches instead of traditional algorithms. (If human experts solve this problem, it is not because they can magically overcome the theoretical limitations of computational complexity. The power of human intelligence in these problems arises from domain-specific knowledge and powerful information processing strategies that characterize intelligence. Together they enable the human to solve particular and important *subsets* of the original problem.) Of course for a number of problems of importance that humans solve, we neither have any kind of algorithms, nor do we have any real idea of how humans solve them. After all, AI as a science is only in its infancy.

3. Intelligence as exploratory process

I would like to propose a view of intelligence as a particular kind of computational process. I hope my description will help in seeing how these processes can be integrated with other kinds of algorithms that also embody human expertise.

Consider an engineer working on a design problem. Let us say she has some paper and pencil and maybe a computer terminal in front of her. Part of the design activity takes place in her head, part of it is recorded in the paper, and part of the needed information processing takes place in the computer by means of execution of some algorithms either she wrote or she invoked. Typically she uses her thought or mental problem solving processes to decompose the problem, think of possible design approaches, previous successful designs which have some similarity to the current problem, visualize the design, do some spatial reasoning or qualitative simulation of the artifact under design, etc., etc. Partial designs are probably set down on the paper, which thus provides an enlarged short term memory capacity. Note that the algorithm executed by the computer is such that it would not be particularly appropriate or possible or efficient for her to do it in her head. She might herself have designed the algorithm, but there is a qualitative difference between the operation of that algorithm and her processes of intelligence.

One way in which to clarify some of the issues is to make a distinction between computations which are *being intelligent* versus those which use the result of earlier intelligent behavior. One might look at an algorithm for the greatest common divisor, and exclaim, "What a clever algorithm!". In reality, the creator of the algorithm was the one who was being clever during its construction, but the algorithm itself, during its running, is not engaging any of the processes that intelligence is composed of, such as exploration of possibilities, hypothesis-making, etc., using general methods for such behavior.

We can focus the discussion by considering what it means to be intelligent in problem solving. Mycin, R1, finite element methods, linear programming algorithms, multiplication algorithms, etc., are all computational methods which provide solutions to some problems. Let us now consider a subclass of methods which are "intelligent" in the following sense: they explore a *problem space*, implicitly defined by a *problem representation*, using general search strategies which exploit typically qualitative heuristic knowledge about the problem domain. A working hypothesis in AI is that humans, unassisted by other computational techniques or paper and pencil, engage in this kind of problem solving. The subarea of AI concerned with problem solving takes as its subject matter the phenomena that surround this kind of knowledge-based and general strategy-directed exploration of problem spaces. The power of these phenomena come from the effective way in which they explore very large problem spaces to make plausible and interesting hypotheses, which can then be verified by other means if necessary. Also, if information that can only be obtained by other kinds of computations are necessary during this kind of exploratory problem solving activity, then these other methods can be invoked, much as an engineer flits between, on one hand, "thinking" about a problem and making intermediate hypotheses, and, on the other hand, writing down some equations to solve before further he or she engages in further exploration.

For lack of a better term, let us call computational methods that are characterized by such knowledge-based and strategy-directed exploration of qualitative problem spaces *problem space exploratory* techniques. Other kinds of computational techniques, let us call *solution algorithms*. These terms are unsatisfactory, but with proper qualifications they will do.

Note that our distinction is some what different from the fairly classical "heuristic" vs "algorithmic" distinction. For example, the algorithm for the Traveling Salesman problem is complex, so a number of programs which approximate the solution by making various assumptions and approximations have come to be called heuristic solutions to the problem. But, these are still solution algorithms according to our definition, albeit without the properties of provable correctness of the solutions given by them, since these algorithms do not, at run time, engage in exploration of the underlying problem space by use of explicit knowledge and general exploration strategies.

There are many problems for which solution algorithms which are not computationally complex are known. Computer science in general and a number other disciplines take as their subject matter the production of solution algorithms for a number of problems of a general or domain-specific nature. Sorting, multiplication, well-defined optimization problems for which linear programming is applicable are of these types. When problems of this type are identified in any domain, there is no reason to engage in problem-space exploratory techniques. Adopting AI-type solutions to these problems

will merely produce solutions which are qualitative and approximate, and, in comparison with the corresponding solution algorithms, the AI methods are likely to be computationally expensive as well. If during diagnostic reasoning one needs to know the exact value of pressure in the reactor chamber, if one has an equation that can be evaluated for it, and if one has all the information needed to evaluate the equation, then it is silly to use problem-space exploratory methods. On the other hand, for a number of problems such as diagnosis and design in the general case, the underlying spaces can be very large, and solution algorithms of limited complexity are not available, except perhaps for some subcases. This is when consideration of AI methods is appropriate.

It is important to emphasize that expert knowledge consists of both kinds of computations. Thus expert systems should use both kinds of techniques, deploying each kind for subproblems where their power is needed. But, since these solution algorithms are domain-specific, or use methods, such as linear programming, that are not the subject matter of AI per se, it is most useful to confine the discussion in AI to problem space exploratory techniques, especially those inspired by human intelligence. For example, I have described elsewhere a study of human-like problem space exploration for the task of design[2].

This view of intelligence as problem space exploration is close to Newell's *problem space hypothesis*[3]. However, in my view an additional characterization of intelligence emerges from a consideration of the nature of exploratory control strategies. In our research we have identified a number of general strategies that we call *generic tasks* to set up and explore problem spaces. I have described them in a number of papers[4,5]. These strategies have the property that they bring an element of computational tractability by using knowledge expressed and organized in specific forms.

4. Concluding Remarks

So far my goal has been to help engineers interested in AI for problem solving and the construction of expert systems to understand what makes AI as a distinct type algorithm by pointing out that a part of human problem solving expertise comes in the form the abilities to explore and search in problem spaces. In this view, the relationship between the thoughts of a problem solver is that they stand for descriptions of the states of a problem space as the problem solver is exploring alternatives in pursuit of a goal. However within AI there is another paradigm about the relationship between thoughts. In this view thoughts are connected by their logical relations and thus "reasoning" is the basic metaphor of artificial intelligence as it applies to deliberative behavior. Different kinds of logics are proposed to capture this relationship and the term "inference" is used to describe the process.

We have used the term "deliberative" several times to make sure that we have been referring to "thinking" as the basic activity of intelligence. However, even within symbolic or algorithmic AI, there have been researchers who emphasize the importance of non-deliberative aspects of intelligence, in particular about phenomena of memory. Minsky and Schank have been noted for theories about organization of knowledge and events in memory structures. The phenomena of memory organization in this research paradigm do not have much to do with reasoning in the sense of logic, nor with problem solving in the sense of search through alternatives that are generated at run-time and examined. Instead they emphasize memory organization and indexing for recognition and retrieval. Sometimes a solution to a problem can be obtained by couching it as a problem for which recognition or retrieval can produce a solution, such as in case-based reasoning.

Thus given a problem to be solved, there are a number of ways in which it may be solvable:

- A "closed form" algorithm may exist for that class of problems, e.g., algorithms for multiplication, sorting, or finite element analysis. If the domain is such that numerical quantities are central to it, then this algorithm will be quantitative in character.
- The solution to the problem may be obtained by generating alternatives in a problem space. This is the *problem solving* view and knowledge, often in a qualitative form, is used to select alternatives that are likely to lie in the path towards a solution.
- The solution may be obtained by logical reasoning from assumptions about the domain, ie., the problem is thought of as a reasoning problem. Some form of theorem proving may be used, and knowledge about the domain is stated in the form of statements in some logical language, most commonly in predicate calculus.
- The problem may be solved by retrieving solutions from memory or transformations of solutions of analogical problems stored in memory. This version of the solution may be implemented in some theories in symbolic forms and in others in a connectionist framework.

Given these possibilities, expertise may come in a form appropriate for any one of the above approaches.

In what follows, I will discuss applications of AI to building problem solving systems, and see what implication the above analysis has to how expertise can be integrated. For simplicity, I will restrict myself to only one type of AI process, namely deliberative goal-seeking, problem space exploration. My comments on integration can be extended to

other types of AI processes as well.

5. Acknowledgments

I acknowledge the support of Defense Advanced Research Projects Agency, contract RADC F30602-85-C-0010, and of Air Force Office of Scientific Research, grant 87-0090, in developing this view of intelligence as a computational process.

6. References

1. B. Chandrasekaran, "What kind of information processing is intelligence? A perspective on AI paradigms and a proposal," to appear in *Foundations of Artificial Intelligence: A Source Book*, D. Partridge and Y. Wilks, editors, Cambridge University Press, 1988.

2. B. Chandrasekaran, "Design: An information processing-level analysis," to appear as Chapter 2 of *Design Problem Solving: Knowledge Structures and Control Strategies*, D. C. Brown and B. Chandrasekaran, forthcoming.

3. A. Newell, "Reasoning, problem solving and decision processes: The problem space as a fundamental category," in R. Nickerson, ed., *Attention & Performance VIII*, Erlbaum, Hillsdale, NJ, 1980.

4. B. Chandrasekaran, "Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design," *IEEE Expert*, Fall 1986, pp. 23- 30.

5. B. Chandrasekaran, "Towards a functional architecture for intelligence based on generic information processing strategies," *Proc. International Joint Conference on AI*, Milan, Italy, August 1987, pp. 1183-1192.

A Design Strategy for Autonomous Systems

Pete Forster

Department of Artificial Intelligence
Edinburgh University
5 Forrest Hill
Edinburgh EH1 2QL
SCOTLAND

Abstract

This paper identifies and puts forward some solutions to crucial issues regarding the competent performance of an autonomously operating robot, namely that of handling multiple and variable data sources containing overlapping information and maintaining coherent operation whilst responding adequately to changes in the environment. Support for the ideas developed for the construction of such behaviour are extracted from speculations in the study of cognitive psychology, an understanding of the behaviour of controlled mechanisms and the development of 'behaviour-based' robots in a few robot research laboratories. The validity of the ideas expressed is supported by some simple simulation experiments in the field of mobile robot navigation and guidance.

1 Introduction

In many robot application scenarios, a human is in close communication with the robot. The human may be closely involved in the execution of tasks or merely have a supervisory role. In either case, malfunction or erroneous operation by the robot can be rapidly observed and hopefully prevented or corrected by the operator. In more remote environments (eg. space and subsea), it may not be possible to utilize human involvement to such an extent due to the limitations of communication. Thus robots operating in such environment must be able to operate autonomously.

What properties does such a robot need to possess and how can they be achieved? The answer to both of these questions is not obvious. The issue of autonomy has already been addressed, but it is not sufficient merely to state that autonomy is the basic requirement for independently operating robots. Future generations of robots will continue to interact with humans, if only to be switched on and off. Robots are effectively the slaves of humans and must be able to receive and react to instructions from them. From this point, the term "autonomous robot" should be taken to mean a robot that is not only able to operate in some autonomous manner, but is also capable of benefiting from information or directives contributed by humans or other sources. Communication channels must not only exist between human and robot, but also between robots and between a robot and any other information sources. Thus an autonomous robot might be summarized as shown in Figure 1.

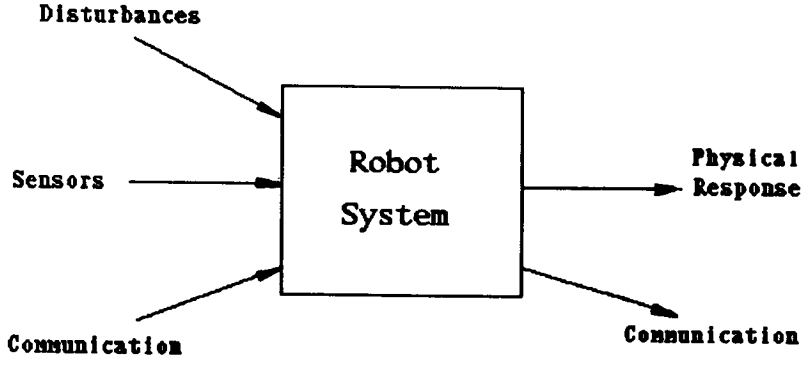


Figure 1: Input-Output schematic for an autonomous robot

Not all inputs to the robot are controllable. The robot may be continually perturbed by the environment; for example, an underwater robot may be subjected to drift by currents and a land-based rover is subject to being rocked or skidding. These inputs are categorized as 'disturbances'. Physical response includes both the response of the robot and its effectors.

2 Some Specifications

Specifying the necessary behaviour for an autonomous robot to operate competently is not only highly domain dependent, but can result in very lengthy top-down breakdowns. This in itself may influence the way a robot's systems are architectured. This paper takes a different angle by considering in more general terms the necessary characteristics of an intelligently acting robot. Intelligence in this context means 'competence in the robot's operating domain'. The criteria under consideration are:

- the robot must be able to operate autonomously;
- the robot should be capable of flexibly utilizing any obtained knowledge which could benefit its performance;
- the robot must be able to respond adequately fast to any scenario that it may face whilst maintaining coherent task strategies;
- the robot should be capable of communicating with and accomodating communications from external sources or agents;

The topic of autonomous operation has already been raised which prompted the issue of how to orchestrate and process inputs from the robot's sensors and other sources with any communication from human sources. Thus the second criteria is specified so that the robot is able to respond as effectively as possible given the available data. This concept allows for degradation and failing of inputs to the robot. It is envisaged that considerable redundancy or overlapping of sensor data will be available to the robot.

A major problem incurred when using conventional sequential computing hardware is that only one process can be executing at a given instant. In real-time control and simulation systems, a scheduler might be used to run different processes on different processing cycles so that a given process is repeated regularly enough to satisfy the update rate requirement of the associated control or simulation process. There is nevertheless a computational limitation to this approach. What is required is specific hardware to run many processes sufficiently fast to meet the needs of the whole system. For example, if a subsea robot had reached a target site and was engaged in task planning activity, it must still maintain vigilance within the environment to avoid falling debris, say, or merely hold its position against variable currents. There should be no reason for its reaction time to deteriorate because it is engaged in simultaneous activity. However, intelligent behaviour is not the result of a package of independently running processes. These processes must not only be configured in such a way that necessary response times can be achieved, but also so that a coherent behaviour emerges for the whole system.

3 System Design

There are two areas of study which contribute directly to the realisation of systems which meet the design criteria specified: the study of animate behaviour and the study of mechanism control.

3.1 Animate behaviour

There is no shortage of autonomously behaving creatures to study in order to gain an understanding of how a machine might be constructed so as to exhibit intelligent behaviour. However, the topic is still surrounded by controversy and leading researchers can still only speculate in the main on how animate behaviour arises. Nevertheless, schemas developed in order to assist us to comprehend the function of biological processing mechanisms can provide useful principles for robot design techniques. Stillings [Stillings et al, 1988] describes human skill behaviour as arising from a combination of *controlled* and *automatic* processes. During the skill learning phase, he interprets subsequent improvement as being a transition of process bias from the slow and methodical controlled process domain to the rapid automatic process domain. He further identifies our limitations in only being able to cope with one 'high-level' controlled process at a time, and the limited control we have over our memory. Computational machines can be designed to overcome these limitations.

Robot processing is conducted on non-biological hardware which suggests that intelligent behaviour should not necessarily be constructed by replicating biological processing, but by determining methods that exploit the characteristics of this more understandable device. There are clearly areas in which processing strategies developed from neurological studies might be appropiate (eg. vision), but believing that all processing could be conducted using similar techniques is optimistic, if not naive.

3.2 Mechanism behaviour

An alternative domain in which to extract concepts for the design of intelligent behaviour is that of mechanisms, particularly those using feedback control processes. Figure 2 outlines phases involved in the response of a system to inputs. The physical system consists of the basic structure of the machine together with the necessary effectors (output mechanisms) with which the machine response may be controlled (eg. steering gear). The physical system is subject to being disturbed by external physical influences such as gusts, rough terrain etc.

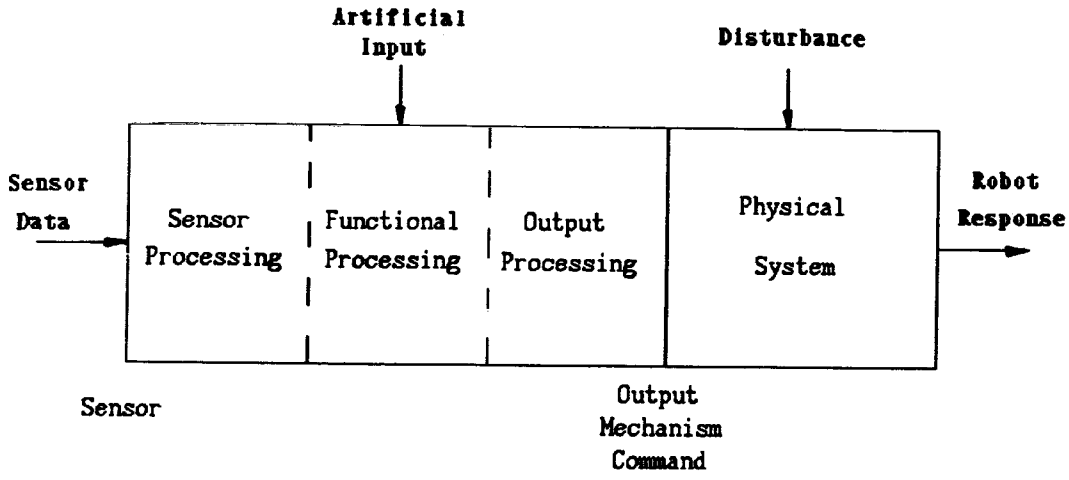


Figure 2: Breakdown of a controlled system

Processing to control the machine is usually conducted on processors, but can be mechanical (eg. directional control of a wind-mill). The processing element issues commands to the effector control devices which modify the response of the whole machine. This processing can be split into three components for most, if not all, controlled mechanism designs. Initially, raw sensor data is processed into a more usable form. This sensor system may govern the way the sensed data is used (as in the *collision avoidance* process described later). This refined data is then used by the functionally oriented processing to produce response demands which can be translated by further output processing into effector demands.

Mechanism processing does not only have to control effectors that directly generate machine response, but they may also have to control the hardware containing the sensors (eg. control of a camera position, orientation, focus and zoom). By controlling the data supplied to the sensor system in this way, this output can be used to indirectly drive the machine response as shown in Figure 3.

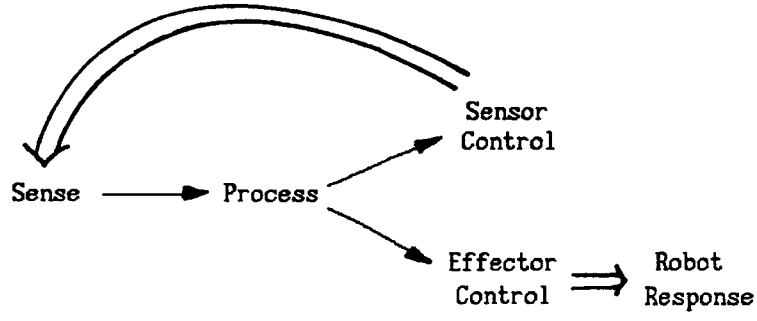


Figure 3: System control paths

3.3 Behaviour-based systems

Several prominent AI researchers have recognized the need to design autonomous systems using *behaviours* as a building block. Brooks [Brooks, 1987] at M.I.T. has designed autonomous mobile robots in this way using a *subsumption* architecture and robust *behaviours* in a distributed processing implementation. Raibert [Raibert, 1986] has successfully built multi-legged robots by commencing his design with a single leg implementation before progressing to the multi-legged case. These approaches attempt to obtain fast and competent response to the environment before incorporating any explicit environment interpretation and reasoning. By orchestrating these reactive behaviours in a coherent manner, more sophisticated competent behaviour may emerge. The problems incurred in the design process are that of specifying what behaviours should be explicitly designed and how they might be orchestrated to construct an intelligent robot or system.

3.4 A processing configuration

Figure 4 describes a processing arrangement to implement an intelligent processing system based on the concepts expressed in the previous three sections. The architecture attempts to ensure that the system responds adequately fast to the environment by executing multiple processes in a distributed fashion. Automatic or subcognitive behaviour lies at the lower end of the diagram where the computation time is short for simple sensors, equivalent to proprioception in animate behaviour, to initiate a response. Types of behaviour found in this category would be collision or hazard avoidance and general reflexive actions. Feeding into these low-level processes are the higher level controlled or cognitive behaviour processes.

Several methods of feeding in to the low-level processes without significantly increasing response times have been investigated. The first method consists of directly combining output response demands (by simple summation or by taking minimums or maximums) after the functional processing stage has been completed. Other methods consist of interfering with the sensor system processing to partially control the behaviour of the low-level processes. This can be performed either by suppressing/enhancing the sensitivity of components of the sensor sytem, or causing 'hallucinations' in the sensor system to evoke particular response. Both of these methods are examples of how a higher-level can exploit the characteristics of lower level processes. Particular examples are described in the example in the following section.

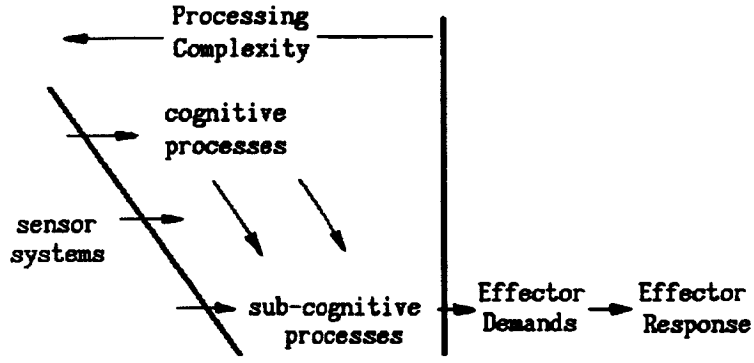


Figure 4: Processing configuration

3.5 A flexible architecture

An autonomous robot is likely to possess a multitude of sensing devices. Each device will provide a unique set of data but the contents of which are likely to be duplicated or part-duplicated by other sensor devices. The quality of the data returned by sensors is likely to vary according to the environment and those sensors may degrade or fail during the operation of the robot. Thus robots should be designed to flexibly utilize any information sources available to them to be able to perform as well as possible within the constraints imposed on them. Methods of amalgamating such sensor derived data have been established [Durrant-Whyte, 1987] [Forster et al, 1988] by manipulating multi-sensor state and estimated variance data. Hallam [Hallam, 1985] has developed self-navigation algorithms which are able to flexibly accomodate various types of data in deriving navigational information. This is described in more detail at the end of the following section.

4 Motion Guidance Example

The field of motion guidance is a prime application area both due to the multiple considerations present and the considerable amount of work already performed in this domain. A *collision avoidance* algorithm has been designed and implemented in a computer simulation for an autonomous vehicle. A simple sensor system consisting of a range finding device of limited azimuth resolution is used to achieve the avoidance behaviour. The algorithm processes data received from this sensor system and produces speed and turn-rate demands. In a wheeled vehicle, this would require the drive power/gearing and steering to be controlled. These are the 'effectors' that will generate vehicle response. The algorithm is notable in that it does not use any memory storage between processing cycles and does not involve any explicit interpretation of the sensor data to produce a map of the sensed environment. Such activity complicates the processing task which leads to an increasing response time. Experiments have shown that the system is capable of exhibiting quite complex motion behaviour. Figure 5 illustrates typical behaviour.

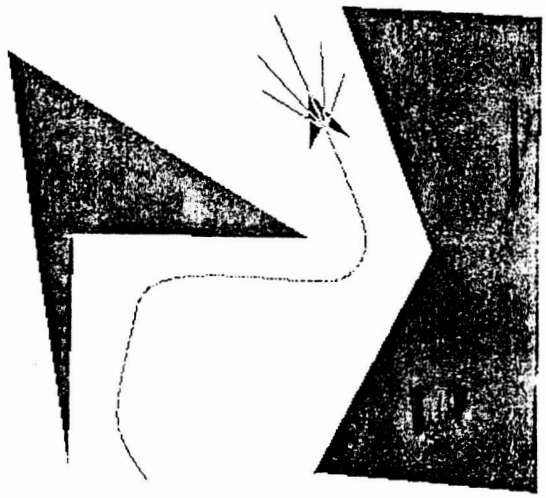


Figure 5: Mobile robot collision avoidance behaviour

The vehicle trail is shown as a dotted line, increasing density indicating a reduction in speed. The sensor system is represented by a number of whiskers protruding from the vehicle. If these whiskers intersect a wall, then the proportion of whisker length obstructed is processed by the algorithm to generate avoidance motion. The number of whiskers and their azimuth spread increases as the sensed environment becomes increasingly cluttered. This will also initiate a reduction in vehicle speed which enables it to perform smaller radius turns.

A simple direction seeking algorithm, based on proportional navigation techniques, has been designed to demonstrate *location seeking* behaviour. This algorithm processes goal direction data and, as for the *collision avoidance* behaviour, produces speed and turn-rate demands. The behaviour consists of the vehicle aiming towards the goal direction. Figure 6 illustrates an example of this behaviour in free space with the vehicle starting from rest, facing in a direction almost 180 degrees away from the goal direction. No range information is provided, so this algorithm results in the vehicle overshooting the goal.

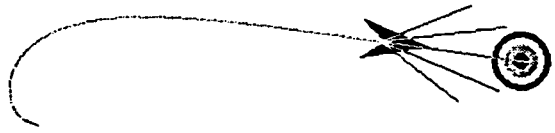


Figure 6: Mobile robot location seeking behaviour

The *location seeking* process is not independent of other processes since it requires direction data. This could be supplied directly from a sensor system (eg. visual) if the location could be sensed. However, a 'higher level' process could formulate a plan with suitable way-points and feed these to the location seeker; this is an example of how a high level process can exploit the capability of a low level process.

The collision avoidance and location seeking processes have potentially conflicting effector output demands. If a resultant competent behaviour is to be achieved, the two processes must be combined so that a single pair of effector output demands are derived. Without further processing of sensed data, the collision avoidance behaviour should maintain dominance over the location seeking behaviour but the resulting behaviour should be seen to take the demands of the location seeker into account whilst the collision avoidance behaviour is stimulated. Changing between processes does not produce a competent behaviour. A successful method of amalgamation has been achieved by modifying the *collision avoidance* algorithm so that its outputs were unbounded and by creating a merging algorithm to combine the two pairs of outputs and bound the result. The merger algorithm simply takes the minimum of the speed demands and the sum of the turn rate demands.This combined structure generates *trajectory guidance* behaviour. An example validation of this behaviour is shown in Figure 7.

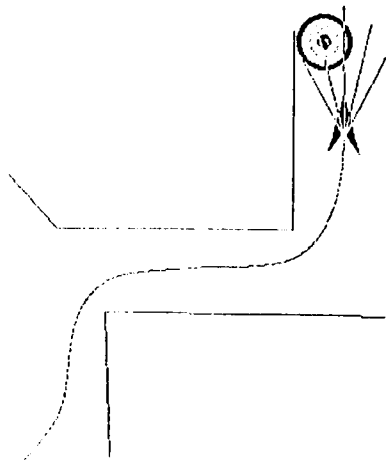


Figure 7: Mobile robot combined motion guidance behaviour

The above two examples of behaviour-based processes are just potential components of a complete autonomous vehicle. More competent behaviour could include the ability to perform navigation and guidance to achieve higher level objectives or tasks. Meeting this particular strategy, a self-navigation framework has been developed [Hallam, 1985] [Forster et al, 1988] to exploit variable type and quality of navigation data and methods. A self-navigation algorithm has been designed and tested in a simulated underwater environment notably making use of sonar derived environment data to infer motion with respect to the environment as well as simultaneously generating a world model. The navigator structure is flexible in that it can further accomodate *a priori* data, dead-reckoning data, navigation data obtained by beacons etc. The data is combined (or fused) by manipulating corresponding measurements and variances using Kalman filter arrangements. The system is robust in that it attempts to perform as well as possible with the data that is available to it.

A possible scheme for the incorporation of this self-navigation process into a general autonomous vehicle system is portrayed in Figure 8. The arrows connecting the boxes represent the general data flow, but the process of exploitation described earlier is less easy to represent. Algorithms generating route guidance strategies may need to exploit not just the collision avoidance process, but also control it by methods of sensor system 'suppression' and 'hallucination'. The simplicity of the collision avoidance behaviour may lead to difficulties in arranging fine motion control in confined locations. If the higher levels are confident about the nature of the environment, then suppression of the sensitivity of the collision avoidance process will lower resistance to motions to be achieved in confined space.

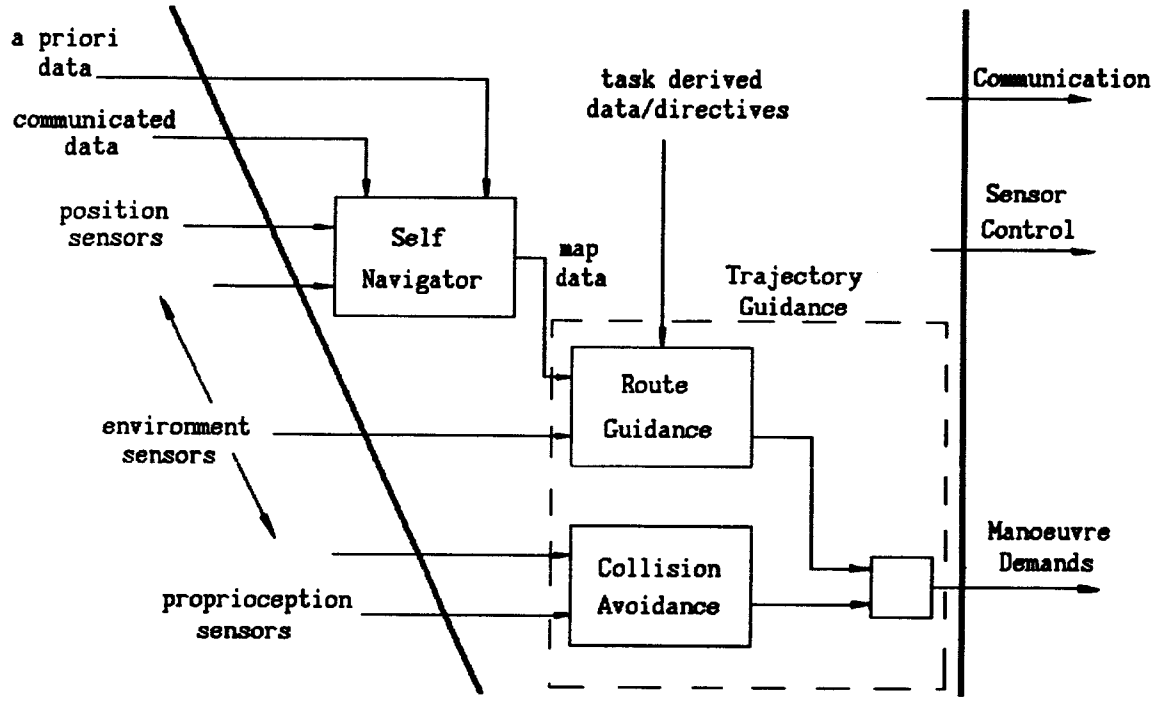


Figure 8: An architecture for mobile robot self-guidance

5 Conclusions

This paper has identified and put forward solutions to crucial issues regarding the competent performance of an autonomously operating robot, namely that of handling multiple and variable data sources containing overlapping information and maintaining coherent operation whilst responding adequately to changes in the environment. The validity of the ideas expressed have been supported by simple simulation experiments in the field of mobile robot navigation and guidance.

6 References

[Brooks, 1986]
Brooks, R, "Achieving Artificial Intelligence through building robots"
M.I.T., A.I. Memo 899, May 1986;

[Durrant-Whyte, 1987]
Durrant-Whyte, H. F., "Uncertain geometry in robotics",
Proc. IEEE Conference on Robotics and Automation, Vol. 2, p851, 1987.

[Forster et al, 1988]
Forster, P, Hallam, J, Smart, J and Howe, J,
"Feature-based navigation for an underwater vehicle",
Proceedings of the 2nd International Workshop on Subsea Robotics, Japan, 1988.

[Hallam, 1985]
Hallam, J C T, "Intelligent automatic interpretation of active marine sonar",
PhD thesis, Edinburgh University, 1985.

[Raibert, 1986]
Raibert, M, "Legged robots that balance", M.I.T. Press, 1986.

[Stillings et al, 1987]
Stillings, N, Feinstein, M, Garfield, J, Rissland, E, Rosenbaum, D, Weisler, S & Baker-Ward, L, "Cognitive Science - An introduction", M.I.T. Press, 1987.

Learning in Tele-autonomous Systems using Soar*

John E. Laird Eric S. Yager Christopher M. Tuck
Michael Hucka
Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109-2110

Abstract

Robo-Soar is a high-level robot arm control system implemented in Soar. Robo-Soar learns to perform simple block manipulation tasks using advice from a human. Following learning, the system is able to perform similar tasks without external guidance. It can also learn to correct its knowledge, using its own problem solving in addition to outside guidance. Robo-Soar corrects its knowledge by accepting advice about relevance of features in its domain, using a unique integration of analytic and empirical learning techniques.

1 Introduction

Tele-robotics allows for intelligent action at a distance. A human, with a long history of solving manipulation and construction tasks, can control a remotely located robot without incurring the risks or costs associated with the environment. However, if the task is repetitive and boring, or if there is sufficient time-delay between the human's instructions and the robot's actions, the robot will have to assume more of the responsibility for intelligent action [5]. By endowing the robot with some intelligence, it should be able to handle routine operations, as well as respond quickly to emergency situations. But how does the robot become intelligent?

Learning through interacting with a human is one way to increase the knowledge of a robot. Initially, a robot may have only very general abilities and must be tightly controlled by a human operator. Through its experiences, the robot can become more and more autonomous. It can increase its repertoire of methods for solving problems, improve its reaction time to events in the environment, and learn to notice new properties of objects in the environment. When a problem is sufficiently novel, so that the robot is unable to easily solve it, the robot can request guidance from a human. The robot can then learn from the interaction so that future intervention is not necessary. If human advice is not readily available, the robot can attempt to solve the problem itself, searching through the space of possible safe actions until it finds a solution. During this problem solving it can learn to avoid actions that do not lead to a solution as well as learn appropriate actions for solving a given class of problems.

Although learning is important in creating intelligent autonomous systems, it can not be considered in isolation, to be grafted onto an existing architecture at a later time. The architecture must be able to attack problems when it has very little knowledge and requires significant outside guidance, as well as when it has large amounts of knowledge and no outside guidance is required. The architecture we use is Soar, an architecture with proven general problem solving and learning capabilities [14]. Soar's learning mechanism, called chunking, is completely integrated with its problem solving so that it learns during problem solving [15].

In this paper we describe initial progress in learning by a tele-autonomous system called Robo-Soar. Robo-Soar performs simple block manipulation task using a Puma robot arm and a machine vision system. We demonstrate the acquisition of new control knowledge so that the system is able to directly solve problems that previously required combinatorial search. We also demonstrate that Robo-Soar is able to learn to correctly manipulate new objects which are not covered by its original knowledge base. This is made possible by

*This research was sponsored by grant NCC2-517 from NASA Ames and ONR grant N00014-88-K-0554.

extending the interface between the human and robot so that the human can point out important features of the environment in addition to providing motor commands.

2 Related Work

Many approaches are possible for learning from experience and outside guidance in tele-autonomous systems. In the simplest case, the human is restricted to controlling the robot's actions, and the learning system must watch "over the shoulder" of the human as the problem is solved. This is the scheme used in robotic programming systems where a human leads the system through a fixed set of commands to achieve a goal. When only the exact commands are stored, the system can only perform that one task because no association is created between the goal and the actions being performed. Another disadvantage is that there is no conditionality in the learned plan. The robot will do exactly the learned set of actions, independent of the state of the environment.

To avoid these problems, "learning apprentices" have been developed that create generalized plans, indexed by the appropriate goal. These systems, such as LEAP [17], are based on a learning strategy called explanation-based learning (EBL) [6,16]. EBL has its roots in the macro-operator learning mechanism of STRIPS [8]. In these systems, an underlying "domain theory" is used to "explain" the actions of the human expert. From the derived explanation, all of the dependencies between the actions are recovered and a general plan is created.

The ARMS system developed by Alberto Segre uses EBL to learn generalized plans for simulated manipulator control [24,23]. The input to the system was the sequence of manipulator moves necessary to construct a specific example of a simple object, such as a revolute joint. Through analysis of the sequence and its own underlying theory of the domain, the system was able to learn general plans that would be independent of the specific example. Boy and Delail [2] have used a similar approach for training a system that advises a human tele-robotic controller.

Our goal is to extend the ARMS approach in the following ways:

1. Solve a problem involving interaction with a real environment. In a real environment, perception is incomplete, so that movements may obscure the current view. In addition, actions and perception are not instantaneous; the system must sometimes wait.

2. Modify the interaction so that the system has more control of the interaction. Therefore, the human will no longer provide a monolithic plan for solving the problem. Instead, the human will give guidance only when the system needs it, or when the human wants. The advantage of this approach is that it minimizes human interaction, and also provides a context for the human's response.

3. Learn individual rules for each decision of the plan instead of learning a complete plan. When plans are learned, it is difficult for the performance system that uses the plans to react quickly to changes in the environment that invalidate the plan. Our goal is to have a system that uses the knowledge from all of its prior experiences for each decision it makes so that, effectively, it dynamically combines portions of independently learned plans.

4. Integrate learning through guidance with general problem solving and autonomous learning. The system should be able to learn from its own experiences, avoiding interaction with humans for simple problems it can solve on its own.

5. Extend the knowledge about the domain. In general, EBL systems are completely dependent on the knowledge encoded in their domain theory. Given the complexity of the world, any finite domain theory for a significant component of the real world will be incomplete and incorrect. We will demonstrate that it is possible to extend an incomplete theory of the domain using a combination of analytic and empirical learning techniques.

Previous work in Soar has already attacked extensions 2, 3 and 4 [9]. In this paper, we demonstrate that the same approach can be extended to real environments where the system's initial knowledge may be incomplete or overgeneral.

We will investigate simple block manipulation tasks. The goal of the robot is to line-up a set of small blocks that have been scattered over the work area. For the first task, all of the blocks are simple cubes that

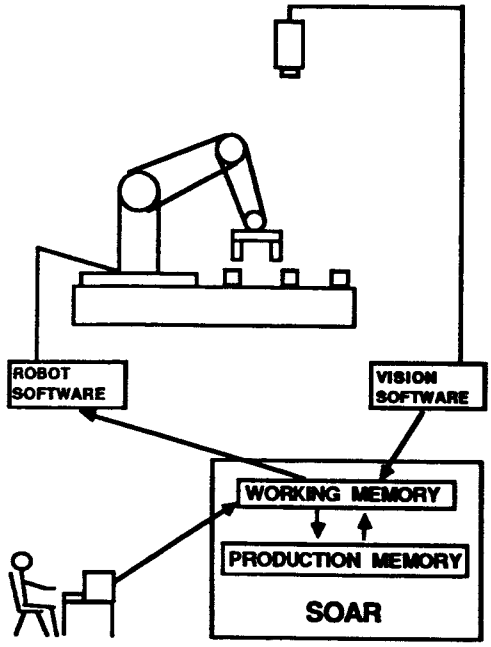


Figure 1: Robo-Soar system architecture.

the gripper can pick up in two different orientations as shown in Figure 1. In the second task, there is a block that is a triangular prism. The gripper is unable to pick up the prism when it closes over the inclined sides. Instead it must be oriented so that it closes over the vertical faces of the block.

3 System Architecture

Figure 1 shows the basic Soar architecture on which Robo-Soar is built.[1] Input comes from a camera mounted directly above the work area. A separate computer processes the images, providing asynchronous input to the rest of the system. The vision processing extracts the positions and orientations of the blocks in the work area as well as distinctive features of the blocks, such as its number. Soar accepts new visual and gripper information whenever it arrives.

In Soar, a task is solved by searching through a problem space of possible states, applying operators to transform an initial state to some desired state. For the block manipulation task, the states are different configurations of the blocks and gripper in the external environment. Some basic operators are shown in the trace of Robo-Soar solving a simple block manipulation problem in Figure 2. These operators correspond directly to commands sent to the robot controller. Robo-Soar solves a problem by selecting operators until it achieves the goal. When an operator is selected, motor commands are sent to the Puma controller and executed. One complication is that the camera is mounted directly above the work area so that the arm obscures the view of a block that is being picked up. Two operators, snap-in and snap-out, move the arm in and out of the work area so that a clear image can be obtained. These operators are necessary, but for simplicity, they will not be included in any of the examples.

This characterization of Robo-Soar does not distinguish it from any other robot controller. What is different is the way Soar makes the decisions to select an operator. Many AI or robotic systems create a plan of actions that the robot must execute. Instead of creating a plan, Soar makes each decision based on a consideration of its long term knowledge, its perception of the environment, and its own internal goals. In this way, it is similar to "situated action" systems [25] such as Pengi [1], and Brooks' Creatures [3]. It represents

[1] Robo-Soar is implemented in Soar 5.0, a new version that allows internal operators to directly modify states [13]. In earlier versions of Soar, operators could only create new states, copying over those aspects of the prior state that did not change.

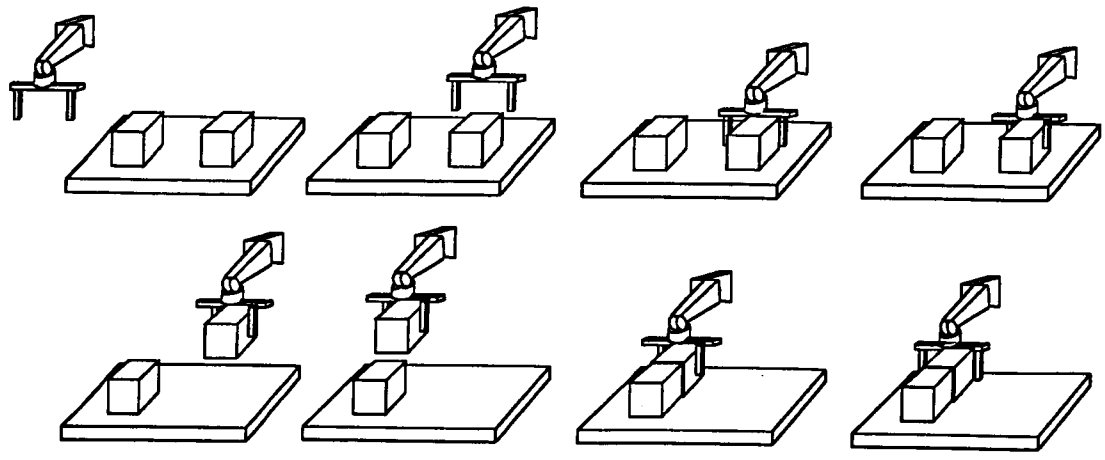


Figure 2: Moving a block using the primitive operators for block manipulation task.

the current situation and its current goals in *working memory*. Soar's long-term knowledge is represented as productions (condition-action rules) that are continually matched against working memory.[2] All of Soar's input and output pass through working memory so that productions can detect changes in the environment, or create expectations which relate to the effects of its actions [27].

In contrast to traditional production systems such as OPS5, Soar fires all successfully matched productions in parallel, allowing them to elaborate the current situation, or create *preferences* for the next action to be taken. There is a fixed preference language that allows productions to assert that operators are acceptable, not acceptable, better than other operators, as good as others and so on. Production firing continues until quiescence is reached (no additional productions match) so that short chains of monotonic inference are possible. Some productions act as bottom-up recognizers, quickly parsing incoming data and building up symbolic descriptions. Other productions compare these descriptions to the goals of the system and create preferences for the alternative operators. Following quiescence, Soar examines the preferences and selects the best operator for the given situation (possibly maintaining the current operator if its actions have not yet completed).

In a familiar domain, Soar's knowledge is adequate to pick and apply an appropriate operator. However, when Soar's preferences do not determine a best choice, or when it is unable to implement the selected operator directly, an *impasse* arises and Soar automatically generates a subgoal. In the subgoal, Soar uses the same approach; it casts the problem of resolving the impasse as a search through a problem space and uses its production memory to control the search when possible. The operators in the subgoal can modify or query the environment, or they may be completely internal, possibly simulating external operators on internal representations. Soar's production memory provides knowledge for selecting operators during the search as well as for implementing the actions of the internal operators. Impasses can arise while selecting or implementing internal operators, so that a hierarchy of subgoals is dynamically created. Within these subgoals, Soar can request advice from a human to help it determine the solution; if advice is not available, it will search.

When Soar creates results in its subgoals, it learns productions, called *chunks*, that summarize the processing that occurred. The actions of a chunk are based on the results of the subgoal. The conditions are based on those working-memory elements that were tested in the subgoal in order to derive the results. This technique is similar to explanation-based learning [6,16]. Although the technique is similar to EBL, most EBL systems learn a plan or schema of the actions in the subgoal. The plan is used to control problem solving in similar goals in the future. Soar's approach is different. The production learned for a goal does not provide control information for future problem solving in that goal. Instead, the chunk encapsulates the processing in the subgoal, eliminating the impasse that gave rise to the subgoal. Control knowledge is learned from subgoals created to select between competing operators.

[2]By exploiting parallelism, advanced production system implementations are very efficient at matching productions [10,11,26].

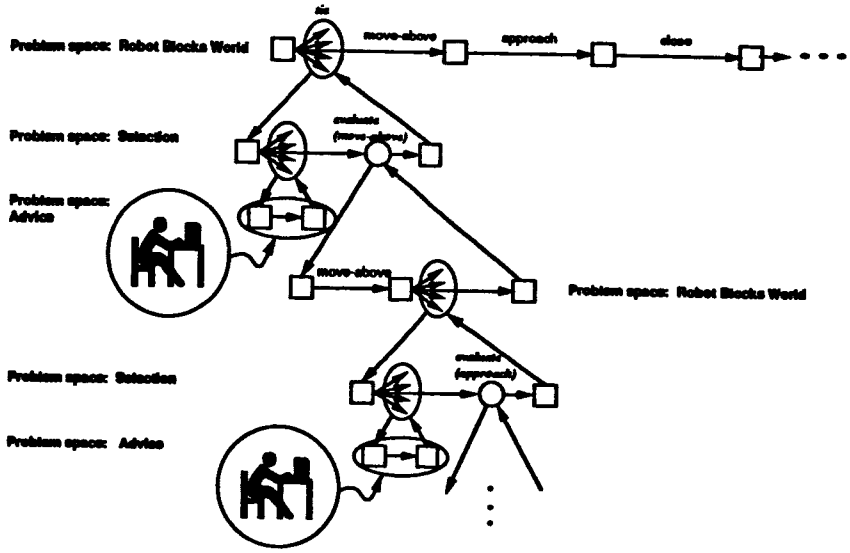


Figure 3: Trace of problem solving using external advice.

4 Learning Control Knowledge from External Advice

To apply Soar to a task, some initial knowledge about the operators in the task must be encoded in Soar. For the block manipulation tasks, this includes productions that suggest operators whenever they can legally be applied, as well as productions that can simulate the actions of the operators internally. In addition there is knowledge about the expected position of blocks following an action. Expectations have to be created because the vision system can not provide positive evidence that an operator applied correctly until it withdraws the arm and snaps it out of the way.

With just this basic knowledge, and no additional control knowledge, Robo-Soar can attempt a task, but it will encounter impasses whenever it tries to select a task operator, as shown in Figure 3. In this figure, horizontal arcs represent operator applications, while squares represent states. Downward pointing arcs are used to represent the creation of subgoals, and upward pointing arcs represent the termination of subgoals. In response to these *tie* impasses, Soar uses the *selection* problem space. Within the selection space, operators are created to evaluate the alternative operators. If the correct operator is evaluated, the system can pursue an internal search to ensure that it is on the path to the goal. Therefore, the efficiency of the search is determined by these decisions. The *advice* is used to obtain guidance from a human. In the advice problem space, the system waits for a human to provide advice. If advice is available, the system uses it to evaluate the appropriate operator. If no advice arrives while the system is waiting, it selects randomly.

Once an operator in the selection problem space has been selected to evaluate a task operator, another subgoal arises if no evaluation is directly available. In response to this impasse, Soar simulates the task operator on an internal copy of the external environment. If the resulting internal state achieves the goal, the operator is evaluated as being "best". If no evaluation is produced, the search continues to determine if another operator can lead Soar to the goal. This results in another impasse, and as should be evident, a depth-first search is performed recursively.

Once the goal is achieved within the internal search, preferences are created to select the operators being evaluated.[3] Each of these preferences is a result, and productions are built to summarize the processing that led to their creation. Figure 4 is an example of the production that is learned for the approach operator. Notice that it not only tests aspects of the current situation, but also aspects of the goal. The productions learned from this search are quite general and do not include any tests of the exact positions or names of the blocks. These features are not included because they were not tested in the subgoal. An important observation about chunking is that the generality of the learning is closely tied to the generality of the goals

[3] If the search leads to illegal or unacceptable states, a preference is created to avoid the responsible operator.

```
If the approach operator is applicable, and
   the gripper is holding nothing, in the safe plane above a block,
   and that block must be moved to achieve the goal,
then create a best preference for the approach operator.
```

Figure 4: Example production learned by Robo-Soar.

and the knowledge used to achieve the goals.

Following the look-ahead search and the accompanied learning, the system is able to directly solve the problem and similar problems with different initial block configurations. It has learned the appropriate operator to apply at each decision point in the search. However, this is not a blind application of a plan. Each of the productions that was learned will test aspects of the environment to insure that they are used only when appropriate. One result of this is that Robo-Soar automatically maintains an operator until sensory information gives it feedback that the situation has changed. In addition, if an operator has an unexpected result, Robo-Soar will not continue with the learned sequence of operators.

5 Refining Incorrect Knowledge

A problem with the traditional learning apprentice approach is that the learning is only as good as the underlying knowledge [20]. If there is an error in the original domain theory, the human has no avenue available for communicating corrections. This is a general problem with deductive learning techniques such as EBL. Although one could argue that errors in the original knowledge can be avoided through careful coding, the same problem can arise when an existing system encounters a problem outside its original specification.

We consider a simple case of this problem by attempting the same task as before except with blocks shaped as triangular prisms. If the original operators were implemented with only cubes in mind, all of the control knowledge and underlying simulation would not be sensitive to the fact that there is another feature in the input that must be attended to. To the Robo-Soar vision system, the prisms look just like cubes, except for a line down the middle at the apex of the triangle. In order to pick up these blocks, the gripper must be aligned with the vertical faces of the block, not just any two sides. If it is not correctly aligned, the gripper will close, but upon withdrawing the gripper, the block will not be picked up.

There are many possible approaches to this problem. First, the system could have an underlying theory of inclined planes, grippers, friction, etc. that it uses to understand why the block was not picked up. One problem with this approach is that this type of knowledge is often difficult to obtain for domains with many "subdomains" [7]. It is hard to know how many of these subdomains are necessary. In addition, the system has to be able to relate its visual input, which in this case is quite limited, with the correct subdomain. A second approach is to gather examples of failure and use inductive learning techniques to hypothesize which feature in the environment was responsible for the problem [18]. This may identify the feature, but it requires many failures and also gives no hint as to the appropriate action. A third approach is for the system to experiment with its available operators to see what actually works [4]. This approach can be quite effective, but it also can be quite time consuming and possibly dangerous. A fourth approach is to open up the robot and reprogram it. This requires skilled programmers and may be difficult if the robot is remotely located. A fifth approach involves increasing the interaction between the human and the robot so that the human can point out relevant features in the environment and associate them with the potential success or failure of a given operator or set of operators.

This latter approach builds on previous work in Soar on recovery from incorrect knowledge [12]. In Soar, recovery is complicated by the fact that chunking is the only learning mechanism, and it only adds knowledge to long-term memory, never modifying existing productions. In our original work, we demonstrated that it was possible to learn new productions that created preferences to correct decisions. The first step in this approach is to notice that an incorrect decision has been made. When Robo-Soar attempts the problem with the prism, the productions it learned on the first task leads it through the first four panels of Figure 5. It correctly moves the gripper above the block. At this point it then approaches the block, closes the gripper,

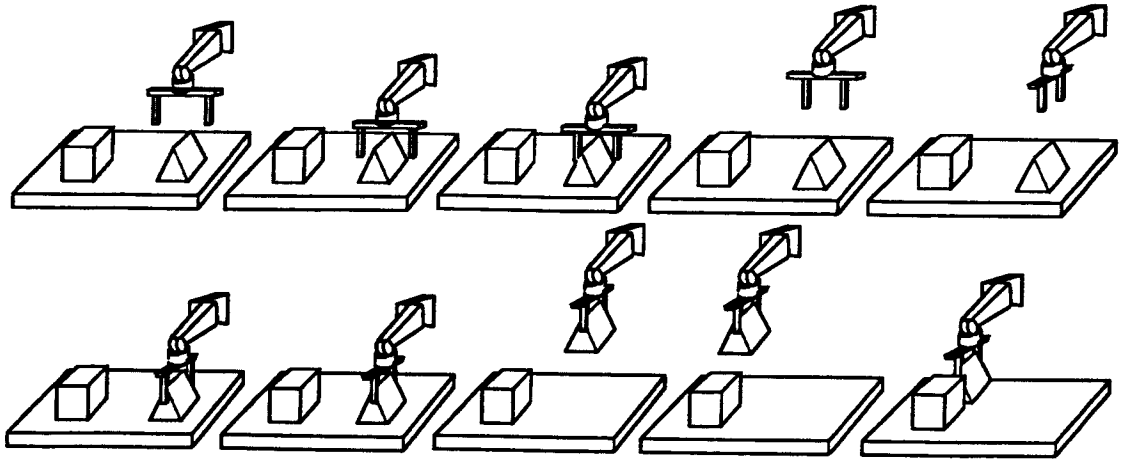


Figure 5: Trace of operator sequence using recovery.

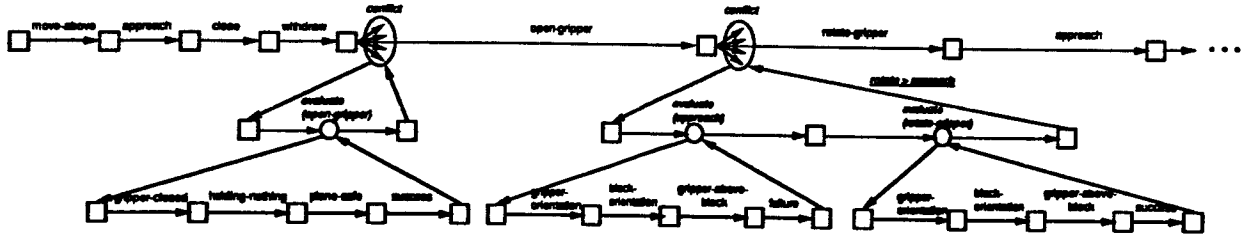


Figure 6: Trace of problem solving with recovery, omitting the advice problem space.

withdraws the gripper and the moves the gripper out of the way of the camera. After this last move it notices that the block is still on the table, so that at some point along the way, an incorrect decision was made.

If no additional knowledge is used, Robo-Soar encounters an impasse because it is in a new situation: the gripper is closed and holding nothing. If we advise it to open the gripper, it will immediately discover (incorrectly) that by approaching the block it will get on the path to the solution. Even though it knows there is an error, it will repeat its attempts to pick up the block indefinitely. This is because its internal knowledge is incorrect. None of its operators, such as approach or close, are sensitive to the orientation of the prism.

To avoid repeating the incorrect actions, domain-independent knowledge is added to Soar that attempts to recover from errors. The general strategy is to not trust the internal knowledge, forcing the system to reconsider each decision and accept guidance from outside. As before, the guidance includes suggestions of appropriate actions, but this time it may include suggestions of inappropriate actions as well as suggestions as to when an inappropriate action should be avoided. The remaining paragraphs of this section go through the details of this approach to recovery.

Reconsideration is implemented by forcing impasses for every decision. These impasses are forced by creating a dummy operator, *deliberate-impasse*, and then creating preferences that make it both better and worse than the other operators. This is guaranteed to force an impasse and allows for the reconsideration of the decision in the resulting subgoal. In our example, the first action to take is to open the gripper. We skip over the process for doing that and go to the situation after the gripper is open as shown in Figure 6. A chunk from the previous problem creates a "best" preference for **approach**. However the conflict impasse prevents its selection and allows reconsideration.

Within the conflict subgoal, the selection space is chosen and internal operators are created to evaluate all

```
If the approach operator is applicable, and
   the gripper is above a block, and the gripper's orientation
   is different from a line in the middle of the block,
   and the rotate operator is available,
then create a preference that rotate is better than approach.
```

Figure 7: Example production learned by Robo-Soar.

of the external operators that were legal for the original state, the same as when an ordinary tie is encountered. At this point the human should not direct the system to the correct action. Even if that is found to lead to the goal, the knowledge to select **approach** still exists. Therefore, **approach** must also be evaluated so that it can be made "worse" than the correct operator, in this case **rotate-gripper**.

Once an evaluation operator for **approach** is selected, any directly computed evaluation is rejected because of a potential for error. Once a subgoal arises to compute the evaluation, a new problem space, called *examine-state* is selected. Its selection is predicated on the fact that an error has been encountered. The purpose of this problem space is to examine features of the state that are relevant to determining the appropriateness of the given operator for the current situation. If just "good" (or "bad") were indicated, the resulting chunks would be overgeneral and always prefer (or avoid) the operator, independent of context. To avoid this, the problem space consists of operators that examine and compare the features of the state, as well as operators that evaluate the state as being on the path to success or failure. Through interactions with a human, the appropriate features are selected and tested, and finally the operator is evaluated as useful or not. If it is deemed to be on the path to success, a "best" preference will be created for it and it will be selected to apply to the top state. However, as is the case for **approach**, failure is selected following the examination of the orientation of the gripper and the block it is above. In this case, **approach** will be avoided and the production in Figure 7 is learned that prevents its selection whenever the gripper is not aligned.

Following the evaluation of **approach**, **rotate-gripper** is evaluated, with the appropriate features of the state being tested before it is judged to be on the path to success. After **rotate-gripper** is selected, another conflict impasse arises, but in this case the human signals that the problem has been fixed and impasse is resolved by rejecting deliberate-impasse. From this point, the chunks apply and take Robo-Soar to the solution. When Robo-Soar encounters future problems, it immediately aligns the gripper before approaching a prism.

6 Discussion

The examine-state problem space is somewhat of a brute-force technique to learn new features. It requires an outside agent to lead the system through a search of potentially relevant features. Although it may not be considered the most elegant or complex machine learning technique, it allows the human to easily correct the system. In addition, this same approach can be used without an outside agent by having the system engage in experimentation. To experiment, the system can guess at relevant features. It will often learn to pay attention to irrelevant features, and thus create overgeneral chunks. But after many interactions with its environment, it will learn to ignore irrelevant features. If this search was a completely blind one, it would be similar to the search through hypothesis space performed by empirical learning techniques. One way to view the examine-state problem space is as an empirical learning method implemented on top of a deductive learning system.

Of course, the search for relevant features does not have to be blind. Many powerful heuristics are available, such as concentrating on new, unknown features, as well as those features that are modified by the operators under consideration. For example, if the system has discovered that the rotate operator is necessary, it could concentrate its search for features relevant to avoiding approach to those modified by rotate. Carbonell and Gill [4] have proposed more deliberate experimentation techniques that would be applicable for this task. Rajamoney and DeJong [19] have described a more elaborate approach to experimentation that is used to learn theories of physical devices.

The goal of our research was to demonstrate the practicality of learning using tele-robotics in a real domain.

Our actual task was quite simple, but Robo-Soar's ability to learn to solve these problems demonstrated the idea. Our current goal is to extend Robo-Soar to more complex tasks, expanding the spectrum of human interaction. At one end, we plan to investigate increasing the modes of communication, so that the user can interrupt Soar at any time to give advice, dynamically reprogramming the system through instructions. One of the original inspirations of the Soar project has always been to create an Instructable Production System where the system is never programmed, only given high-level advice [21,22]. We will also expand the current interface, so that it is more natural for the human to pick actions and relevant features. On the other end of the spectrum, we plan to study experimentation techniques so that Robo-Soar will be able to learn much of the same information on its own, when human advice is unavailable.

7 Acknowledgments

We would like to thank Karen McMahon for implementing Soar 5.0, and Mark Wiesmeyer for developing and implementing the Soar input and output interfaces. Without these extension to Soar, Robo-Soar would not have been possible. We also like to thank the staff of the Robotics Laboratory for help with the tele-autonomous software and hardware. Finally, we thank Paul Rosenbloom and Allen Newell for ideas relating to this work.

References

[1] P. E. Agree and D. Chapman. Pengi: an implementation of a theory of activity. In *Proceedings of AAAI-87*, 1987.

[2] G. A. Boy and M. Delail. Knowledge acquisition by specialization/structuration: a space telemanipulation application. In *Workshop on Integration of Knowledge Acquisition and Performance Systems*, August 1988.

[3] R. A. Brooks. Intelligence without representation. In *Proceedings of the Workshop on the Foundations of Artificial Intelligence*, MIT, June 1987.

[4] J. C. Carbonell and Y. Gil. Learning by experimentation. In Pat Langley, editor, *Proceedings of the Fourth International Workshop on Machine Learning*, pages 256–266, 1987.

[5] L. Conway, R. Volz, and M. Walker. Tele-autonomous systems: methods and architectures for intermingling autonomous and telerobotic technology. In *Proceedings of the IEEE International Conference on Robotics and Automation*, February 1987.

[6] G. DeJong and R. Mooney. Explanation-based learning: an alternative view. *Machine Learning*, 1(2):145–176, 1986.

[7] R. Doyle. Constructing and refining causal explanations from an inconsistent domain theory. In *Proceedings of AAAI-86*, Morgan Kaufmann, 1986.

[8] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[9] A. Golding, P. S. Rosenbloom, and J. E. Laird. Learning general search control from outside guidance. In *Proceedings of IJCAI-87*, Milano, Italy, August 1987.

[10] A. Gupta, M. Tambe, D. Kalp, C. L. Forgy, and A. Newell. Parallel implementation of OPS5 on the encore multiprocessor: results and analysis. *International Journal of Parallel Programming*, 17(2), 1988.

[11] W. Harvey, D. Kalp, M. Tambe, A. Acharya, D. McKeown, and A. Newell. Measuring the effectiveness of task-level parallelism for high-level vision. Computer Science Department, Carnegie Mellon University, In preparation, December, 1988.

[12] J. E. Laird. Recovery from incorrect knowledge in Soar. In *Proceedings of the National Conference on Artificial Intelligence*, August 1988.

[13] J. E. Laird and K. A. McMahon. Destructive state modification in soar, draft iv. 1989. Artificial Intelligence Laboratory, The University of Michigan, unpublished.

[14] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33(3), 1987.

[15] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in Soar: the anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[16] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, 1, 1986.

[17] T. M. Mitchell, S. Mahadevan, and L. I. Steinberg. LEAP: a learning apprentice for VLSI design. In *Proceedings of IJCAI-85*, pages 616–623, Los Angeles, CA, August 1985.

[18] M. Pazzani, M. Dyer, and M. Flowers. The roel of prior causal theories in generalization. In *Proceedings of AAAI-86*, American Association for Artificial Intelligence, Morgan Kaufmann, Philadelphia, PA, 1986.

[19] S. Rajamoney and G. DeJong. Active explanation reduction: an approach to the multiple explanations problem. In J. Laird, editor, *Proceedings of Fifth International Conference on Machine Learning*, pages 242–255, Morgan Kaufmann, Ann Arbor, MI, 1988.

[20] S. Rajamoney and G. DeJong. The classification, detection and handling of imperfect theory problems. In *Proceedings of IJCAI-87*, pages 205–207, Morgan Kaufmann, Milano, Italy, 1987.

[21] M. D. Rychener. Approaches to knowledge acquisition: the instructable production system project. In *Proceedings of the First Annual National Conference on Artificial Intelligence*, pages 228–230, AAAI, August 1980.

[22] M. D. Rychener. The instructable production system: a retrospective analysis. In *Machine Learning: An artificial intelligence approach*, Tioga, Palo Alto, CA, 1983.

[23] A. M. Segre. *Explanation-Based Learning of Generalized Robot Assembly Plans.* PhD thesis, University of Illinois at Urbana-Champaign, 1987.

[24] A. M. Segre. Explanation-based manipulator learning. In *Proceedings of the Third International Machine Learning Workshop*, pages 183–185, Skytop, PA, 1985.

[25] L. Suchman. *Plans and Situated Action.* Cambridge University Press, 1987.

[26] M. Tambe, D. Kalp, A. Gupta, C.L. Forgy, B.G. Milnes, and A. Newell. Soar/PSM-E: investigating match parallelism in a learning production system. In *Proceedings of the ACM/SIGPLAN Symposium on Parallel Programming: Experience with applications, languages, and systems*, pages 146–160, July 1988.

[27] M. Wiesmeyer. Soar I/O reference manual, version 2. 1988. Artificial Intelligence Laboratory, The University of Michigan, unpublished.

DESIGN OF A STRUCTURAL AND FUNCTIONAL HIERARCHY FOR PLANNING AND CONTROL OF TELEROBOTIC SYSTEMS

Levent Acar

Dept. of Electrical Engineering
University of Missouri-Rolla
112 Electrical Engineering
Rolla, MO 65401

Ümit Özgüner

Dept. of Electrical Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210

Abstract

Hierarchical structures offer numerous advantages over conventional structures for the control of telerobotic systems. A hierarchically organized system can be controlled via undetailed task assignments and can easily adapt to changing circumstances. The distributed and modular structure of these systems also enables fast response needed in most telerobotic applications. On the other hand, most of the hierarchical structures proposed in the literature are based on functional properties of a system. These structures work best for a few given functions of a large class of systems. In telerobotic applications, all functions of a single system needed to be explored. This approach requires a hierarchical organization based on physical properties of a system. In this paper, such a hierarchical organization is introduced. The decomposition, organization and the control of the hierarchical structure are considered, and a system with two robot arms and a camera is presented as an example.

1. Introduction

In most telerobotic applications, the need to express undetailed tasks and to expect the robotic system to plan and reason about its environment is essential. Hierarchical structures provide systematical methods for planning detailed task assignments and deriving fine-motion control strategies.

There are many other reasons to create hierarchical organizations for telerobotics systems. Most importantly, these organizations help to simplify the controller design by allowing designs on smaller portions of a large, complex system. They also offer distributed computation capabilities and enable local reasoning and planning which are necessary for fast response and error recovery.

Hierarchical structures are based on different representations of a system in Control and in Artificial Intelligence (AI) Theory. In Control, a system is usually defined by its dynamic equations. As a result, most hierarchical distributions in Control exploit mathematical properties in the description of the system. These decompositions range from determining coupling parameters inside system dynamics to input-output correlations of a system, [1]–[7]. On the other hand in AI, organizations are based on functional behavior and accomplishable

goals, [8, 9]. These goals are usually divided into simpler subgoals which form the levels of the hierarchy. Similar hierarchies were also utilized for computational and design purposes, [10]–[12].

To plan control strategies and execute them, the two type of representatives should be unified. In the literature, this unification was addressed in two different approaches. The first approach was a fixed level hierarchy advocated by Albus et al. [13]-[16] and others [17]. Albus used a five level hierarchy where the system is connected to the structure at the last level. This approach was successfully implemented for the control of a robot arm, but it is not general enough to include more complicated systems with numerous functional behaviors. The second method was by Saridis et al. [18]-[20] and it used a three level classification hierarchy with organizational, coordination and executional levels. Similar to the first approach, the system was connected to the hierarchy at the last level. The coordination level could be viewed as a translator between the functional and the mathematical representations describing the system. This hierarchy is more general, because it was explained with vague descriptions, such as the intelligence decreases as the detail and granularity increases. Similar organizations under different names were also given in [21]-[23].

There are two major similarities between the two organizations. Both of them form tree structured hierarchies. However, as pointed out in [24], most of the hierarchical structures have more complicated connections, and levels of hierarchy may not be obvious. Both of these organizations work well for a wide range of systems provided only a few functional behaviors are expected. On the other hand, most telerobotic applications require a system to work well for a variety of functions.

In this paper, a structural and functional hierarchy is proposed to overcome these problems and to obtain an alternative structure. The hierarchy is first formed by considering the physical properties of a system. Then, functions are associated with this organization. Finally, a uniform control flow is described for the hierarchy. A two robot and a camera system is presented as an example to demonstrate the details and the applicability of the structure.

2. Hierarchical Structure and Control

In this section, we introduce the general form of a hierarchical organization for the control of telerobotic systems. The structure consists of functional abstractions associated with a physical decomposition of the system.

We start the decomposition by separating the system into an initial set of components. These components don't have to be disjoint or complete, but as we will observe later, the detail of the hierarchy depends on this initial choice.

The second step will be to obtain the largest set of disjoint components from the initial set. This set of disjoint components will form the bottom level of the hierarchy and the whole system will form the top level. In between these two levels, there are numerous components connected as a directed graph with no structural loop. These mid-structure components must be connected so that there is always a commom portion between a node and any of its

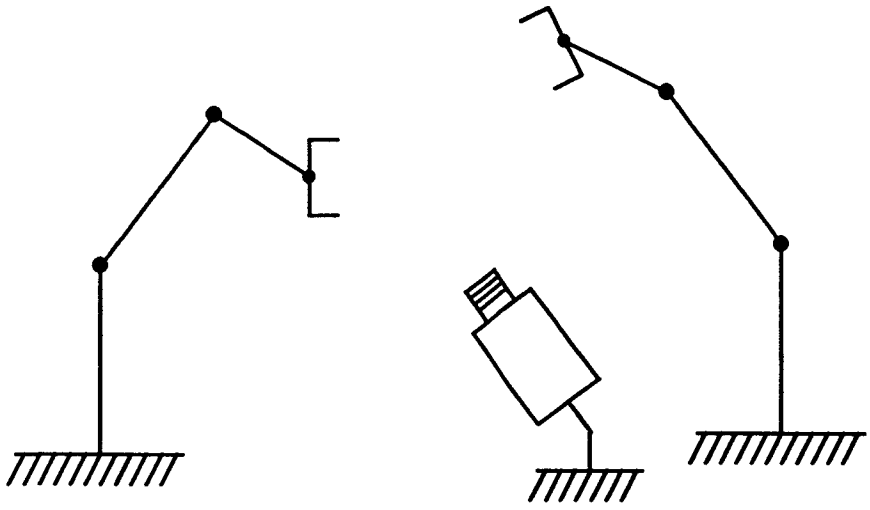


Figure 1: A two robot and one camera system.

subnodes[1], and the collection of all the subnodes of a node contains at least the node itself. These requirements produce a very interesting hierarchical structure where the whole system is represented almost completely at different levels of detail within the hierarchical organization. The top level is the least and the bottom level is the most detailed level. Although, the levels in the middle are not necessarily well defined, selective collections of components from the mid-structure would describe the system at different details.

To see an example of a hierarchical organization, we consider a system with two robot arms and one camera as in Figure 1. One possible hierarchical organization is given in Figure 2, where the dashed boxes represent the initial set of components.

After the formation of a hierarchical organization, functions related to the components are assigned and the flow of the control process is described. To represent the functions and the control process, we introduce six primitives. The primitives exist at every node of the hierarchy and related to each other through the control process. These primitives are:

Goals are assertions representing the end result to be obtained.

Tasks are elementary job descriptions.

Procedures are methods of accomplishing tasks. Procedures are separated into two groups:

1. *Current Procedures* which are locally applicable procedures, and
2. *Subprocedures* which are applicable only by subnodes.

Measurements are the available information from sensors. The measurements are also separated into two groups:

1. *Current Measurements* which are the measurements available locally, and
2. *Other Measurements* which are the measurements of other nodes.

[1]subnodes are sometimes called children nodes

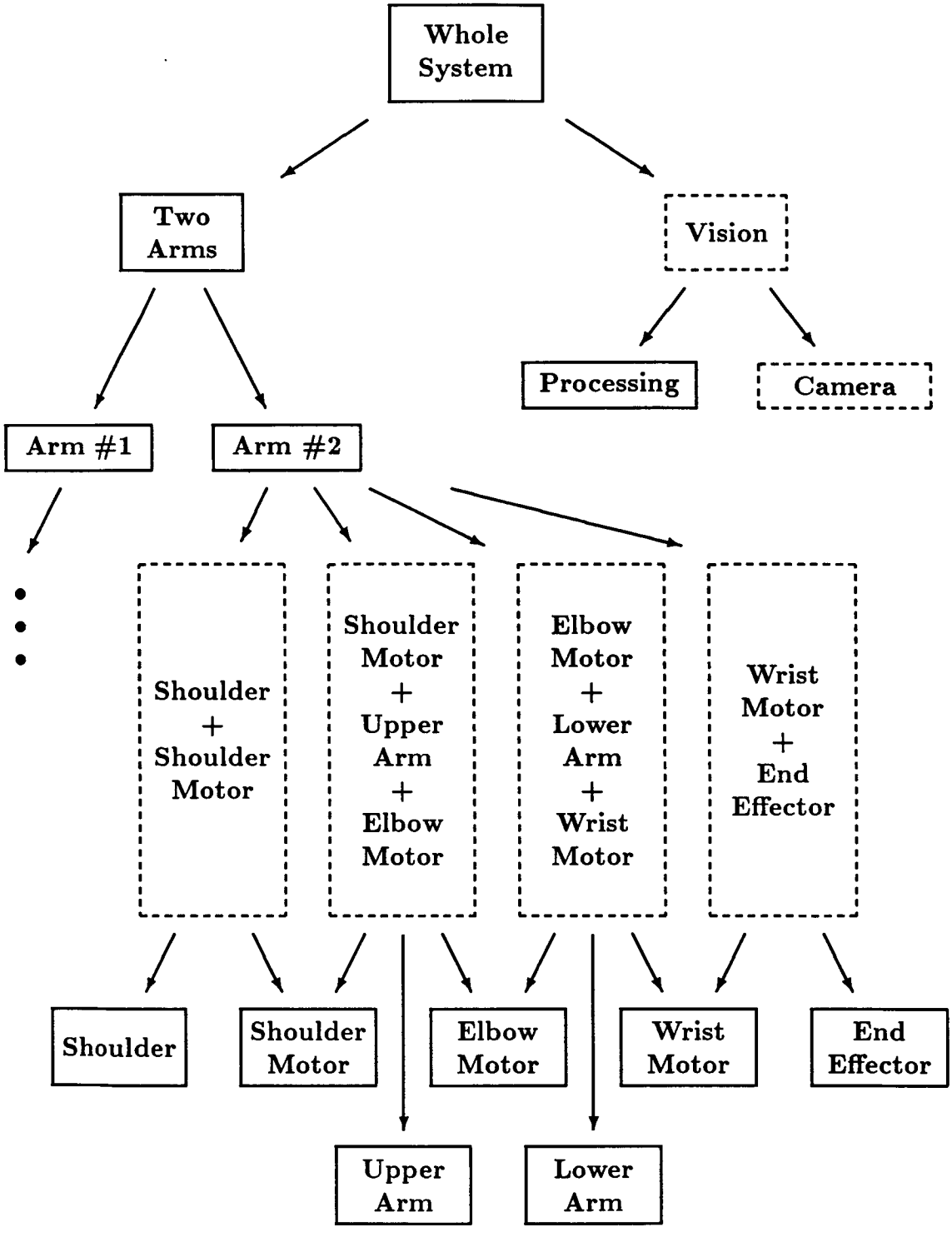


Figure 2: One possible hierarchical structure of the two robot and one camera system.

Constraints are task restrictions or exemptions which are formed by:

1. *System Dependent Constraints* which depend solely on the system and do not change with the environment or the goals, such as angular limitations of a robot arm due to its construction, and
2. *System Independent Constraints* which does not depend on the system, but depend on the environment or on the assigned goals, such as angular limitations of a robot arm due to an obstacle.

Resources are task restrictions or exemptions which are related to the use of procedures. Similar to measurements, resources are also separated into two groups:

1. *Current Resources* and
2. *Other Resources.*

We represent functions of the components by collections of tasks. At each node, a list of tasks with associated procedures, constraints and measurements forms the knowledge base of the node. Procedures associated with the same task provide different ways of accomplishing the task. Utilizing a current procedure implies that the associated task can be accomplished at that node and there is no need to propogate the task further down in the hierarchy. On the other hand, utilizing a subprocedure implies goals have to be formed for the subnodes to accomplish the task. This information provided by the subprocedures is important for timely propogation of tasks in the hierarchy.

In the proposed organization, a procedure may accomplish more than one task and a set of procedures may accomplish a single task. If we need to order tasks and procedures sequentially, additional constraints are included to synchronize the execution.

Procedures also use measurements and consume or produce resources. We include the information about some of the measurements available at other nodes as part of their knowledge bases. This information helps to utilize other measurements directly without the use of usual backward and forward search methods. The knowledge about the resources of other nodes is included for failure handling purposes only, [25], and it is not used for control.

Constraints are also ranked among themselves. We assign a *stiffness constant* to each constraint according to its importance. For example, a constraint on the allowable grasping tension for a robot end-effector can be relaxed if the robot is holding a steel pipe, but it has to be observed strictly if the robot is holding a fragile vase.

We also assign another type of constant for resources and measurements to represent the cost of using them. These costs affect the choice of procedures which use measurements and resources.

The six primitives are related to each other in a unique way by the control process. Every node has the identical process as shown in Figure 3. When a goal with constraints and resources is received by a node, it is first decomposed into a number of tasks such that the accomplishment of the tasks implies the accomplishment of the goal. Since many procedures may be assigned to a task, one set is selected according to a cost criterion. The cost may be energy, entropy or currency as long as an optimum exists. Among the selected procedures,

some may also be subprocedures. In the last stage, these subprocedures are used to form goals for the subnodes. The stiffness constants of the corresponding tasks are also propogated with the goals.

Most of the knowledge is embedded into the system before the system starts execution. The dashed boxes in Figure 3 show the portion of the process which runs in real-time after goal assignments.

The connections of the hierarchy are initially determined before the task assignments. These connections form a static structure. When the system starts running, nodes can be temporarily connected with each other to exchange measurements. This exchange introduces a dynamic structure which may create loops in the organization. However, these loops don't cause any cyclic behavior, since only measurements can be exchanged via the new connections. Without this dynamic structure, the controlling ability of the organization would have been severely limited.

3. Functional Assignments

Next, we will briefly discuss task assignments and goal propogation process for a two robot arm and one camera system introduced earlier. We will only consider a few of the applicable tasks to give a general view of the functional behavior of the structure.

We assume that the system is asked to carry an object from one end of a table to the other end, where both arms have to be utilized. At the top level, the goal *Carry* is matched with the task *Carry*. We assume that there are three procedures associated with task *Carry*: *Carry_alone*, *Carry_separately* and *Carry_together*. Procedure *Carry_alone* utilizes only one arm. Procedure *Carry_separately* picks the object with one arm and puts it on the table between the two arms for the other arm to carry it further away. *Carry_together* transfer the object in the air without putting it on the table. With the help of the *Vision* node which determines distances between locations, the top node decides to use the *Carry_together* procedure. Since all these procedures are subprocedures, a goal for the *Two_Arms* node is formed and decomposed into the following tasks:

GOAL	**TASK DECOMPOSITIONS**
Carry_together ⟶	*Reach* ?stiffness Arm#1 Location_Object
	Lift ?stiffness Arm#1 Location_Object
	Reach ?stiffness Arm#1 Midlocation_1
	Reach ?stiffness Arm#2 Midlocation_2
	Transfer ?stiffness Arm#1 Arm#2
	Reach ?stiffness Arm#1 Location_final
	Reach ?stiffness Arm#2 Destination
	Putdown ?stiffness Arm#2 Object Destination

Constraints are also added to these tasks to preserve the synchronization and the sequential order. The status of the constraints are obtained from the *Vision* node. In the next stage,

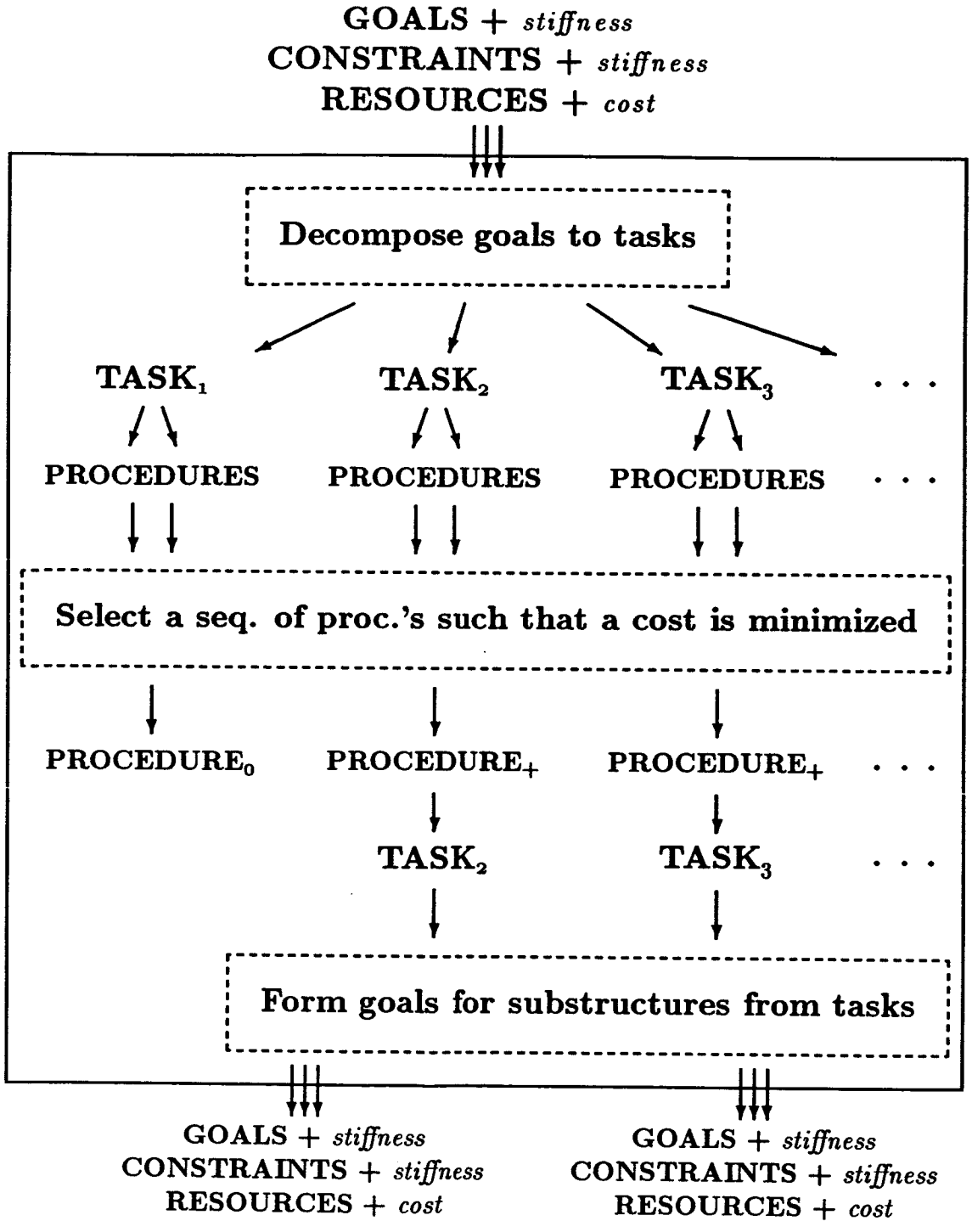


Figure 3: Control flow diagram for each node.

goals for the *Arm#1* and *Arm#2* nodes are formed from these tasks. After a similar decomposition, elementary tasks such as *Move_Coarse*, *Move_Detailed*, *Orient*, *Grasp* and *Ungrasp* are formed. This process continues down to the bottom level nodes where the tasks and procedures are more elementary. Changing joint angles, opening or closing the end-effector and the input voltage profiles are among the tasks of these nodes.

4. Conclusions

In this paper, we considered a hierarchical organization primarily based on the physical properties of a system which is most suited for telerobotics applications.

We formed the hierarchy as a directed graph with no structural loop from the components of the system. We arranged the components so that the system is described with least detail at the top level and with most detail at the bottom level. In between these two levels, the system can be described with any desired detail depending on an initial choice of components. We also did not limit the type of the hierarchy[2]. We deliberately allowed representational redundancies in the middle levels, but we forced the top and the bottom levels to be represented without redundancies. This restriction is to enforce a consistent control at the bottom level.

Then, we assigned knowledge and functionality to the nodes of the hierarchy by using six primitives: Goals, Tasks, Procedures, Measurements, Constants and Resources. These primitives contain the knowledge about the capabilities of the components and the information about their behavior.

Finally, we described the control process as decomposition of goals into tasks, selection of procedures for these tasks and formation of new goals for the subnodes. This process starts at the top level and propogates down uniformly. As control decisions are made at the nodes of the hierarchy, we allowed new connections among nodes for data exchange. These connections save time and make efficient use of all possible control strategies.

To observe the formation of the proposed hierarchy and the propogation of the control process, we considered a system with two robot arms and a camera.

References

[1] M. Jamshidi. *Large-Scale Systems.* North-Holland, Amsterdam, Netherlands, 1983.

[2] F. Harary, R. Z. Norman, and D. Cartwright. *Structural Models: An Introduction to the Theory of Directed Graphs.* John Wiley & Sons, Inc., New York, New York, 1965.

[3] D. P. Looze and N. R. Sandell, Jr. Hierarchical control of weakly-coupled systems. *Automatica*, 18(4):467–471, July 1982. *Brief Paper.*

[2]In the literature, most hierarchies are limited to the tree structured hierarchies.

[4] A. Vannelli. *Solution Techniques for 0-1 Indefinite Quadratic Problems with Applications to Decomposition.* PhD thesis, Department of Electrical Engineering, University of Waterloo, Waterloo, Canada, 1983.

[5] M. E. Sezer and D. D. Šiljak. Nested ϵ-decompositions and clustring of complex systems. *Automatica*, 22(3):321–331, May 1986.

[6] A. H. Nayfeh. *Perturbation Methods.* John Wiley & Sons, Inc., New York, New York, 1973.

[7] V. I. Utkin, S. V. Drakunov, D. E. Izosimov, A. G. Lukyanov, and V. A. Utkin. A hierarchical principle of the control system decomposition based on motion separation. In *Preprints of the 9th World Congress of the International Federation of Automatic Control: Volume V*, pages 134–139, Budapest, Hungary, July 1984.

[8] P. H. Winston. *Artificial Intelligence.* Addison-Wesley Publishing Company, Inc, Reading, Massachusetts, 2nd edition, July 1984.

[9] E. Rich. *Artificial Intelligence.* McGraw-Hill Book Company, New York, New York, 1983.

[10] D. L. Zeltzer. *Representation and Control of Three Dimensional Computer Animated Figures.* PhD thesis, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio, August 1984.

[11] D. C. Brown and B. Chandrasekaran. Knowledge and control for a mechanical design expert system. *IEEE Computer Magazine*, 19(7):92–100, July 1986.

[12] D. C. Brown. *Expert Systems for Design Problem-Solving using Design Refinement with Plan Selection and Redesign.* PhD thesis, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio, 1984.

[13] J. S. Albus, A. J. Barbera, and M. L. Fitzgerald. Hierarchical control for sensory interactive robots. In *Proceedings of the 11th International Symposium on Industrial Robots*, pages 497–505, Japan, October 1981.

[14] J. S. Albus, A. J. Barbera, and M. L. Fitzgerald. Programming a hierarchical robot control system. In *Proceedings of the 6th International Robot Technology*, pages 505–517, Paris, France, June 1982.

[15] J. S. Albus, C. R. McLean, A. J. Barbera, and M. L. Fitzgerald. Hierarchical control for robots and teleoperators. In *Proceedings of the IEEE Workshop on Intelligent Control*, pages 39–49, Troy, New York, August 1985.

[16] J. S. Albus, R. Lumia, and H. McCain. Hierarchical control of intelligent machines applied to space station telerobots. In *Proceedings of the Workshop on Space Telerobotics: Volume I*, pages 155–165, Pasadena, California, July 1987. Jet Propulsion Laboratory.

[17] J. Rasmussen. The role of hierarchical knowledge representation in decisionmaking and system management. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(2):234–243, March/April 1985.

[18] G. N. Saridis. Foundations of the theory of intelligence controls. In *Proceedings of the IEEE Workshop on Intelligent Control*, pages 23–28, Troy, New York, August 1985.

[19] G. N. Saridis and K. P. Valavanis. Software and hardware for intelligent robots. In *Proceedings of the Workshop on Space Telerobotics: Volume I*, pages 241–249, Pasadena, California, July 1987. Jet Propulsion Laboratory.

[20] K. P. Valavanis. *A Mathematical Formulation for the Analytical Design of Intelligent Machines.* PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 1986.

[21] A. Meystel. Nested hierarchical controller with partial autonomy. In *Proceedings of the Workshop on Space Telerobotics: Volume I*, pages 251–270, Pasadena, California, July 1987. Jet Propulsion Laboratory.

[22] J. Jiang and R. Doraiswami. Performance monitoring in expert control systems. In *Preprints of the 10th World Congress on Automatic Control: Volume 6*, pages 303–307, Munich, Federal Republic of Germany, July 1987.

[23] W. J. Wolfe and S. D. Raney. Distributed intelligence for supervisory control. In *Proceedings of the Workshop on Space Telerobotics: Volume I*, pages 139–148, Pasadena, California, July 1987. Jet Propulsion Laboratory.

[24] H. A. Simon. *The Sciences of the Artificial.* The MIT Press, Cambridge, Massachusetts, 2nd edition, 1981.

[25] L. Acar and J. R. Josephson. Hierarchically distributed real-time replanning and plan execution with specialized expert systems. *(In preparation)*, 1989.

NASA GODDARD SPACE FLIGHT CENTER

The Flight Telerobotic Servicer Project: A Technical Overview

Harry G. McCain

NASA Goddard Space Flight Center
Code 409
Greenbelt, Maryland 20771, U.S.A.

Abstract

The Flight Telerobotic Servicer (FTS) Project is developing an advanced telerobotic system to assist in and reduce crew extravehicular activity (EVA) for Space Station Freedom. The FTS will provide a telerobotics capability to the Freedom Station in the early assembly phases of the program and will be employed for assembly, maintenance, servicing, and inspection throughout the lifetime of Freedom Station. The FTS will also be capable of operation on the Orbital Maneuvering Vehicle (OMV) for remote servicing activities. A planned evolution of the FTS capabilities will take place over time with technology transfer between U.S. industry, universities, and other Government agencies as an integral part of the program.

The FTS technical challenge is the development and integration of a spaceflight quality system with both teleoperation and autonomous capabilities. The system must be safe and operate reliably in the space environment. The technical output of the FTS development process will not primarily come from the development of new sensors, manipulator actuator mechanisms or other robotic components. The state of the art in these components is already highly sophisticated. The major contribution of the FTS to the advancement of the state of the art will come from the integration of these components into an operational system which meets the broad spectrum of Space Station Freedom requirements. This paper will provide background and rationale leading to the desired FTS telerobotic capabilities and then describe some of the specific technical requirements to which the FTS must be designed in order to meet these goals as well as operate effectively in the space environment.

1.0 Introduction

Robotic technology must make major advances beyond the current state of the art to even approach human capabilities in areas such as vision processing and task planning. In acknowledgment

of these limitations, the FTS must be designed with sophisticated teleoperation capabilities to keep the human in the control loop whenever there is a requirement for tasks to be accomplished in an unstructured environment. However, it is also required that the initial FTS system have the capability to perform autonomously when the task environment can be sufficiently structured. The evolution of the FTS will allow autonomous operation in less and less structured environments.

2.0 Teleoperated Systems

Manipulator systems which employ teleoperation techniques have been developed by the nuclear industry for the repair and maintenance of nuclear reactors and by the undersea industry for offshore gas and oil rig assembly, repair and maintenance, as well as for undersea salvage operations. Teleoperation allows the operator to accomplish work at a relatively unknown or unstructured worksite in a hostile environment such as the sea floor or inside a nuclear reactor while he or she remains in a safe, shirt sleeve environment.

2.1 Hand Controllers

The primary feature of a teleoperated system is that a human operator is directly responsible for all motion of the manipulator or manipulators at the worksite. He or she is situated remote from the actual manipulator task operations at a workstation. The operator directs the motions of the "slave" manipulator(s) via some form of hand controller device. These devices have several configurations depending on the system.

Systems such as the Space Shuttle Remote Manipulator System (RMS) employ joystick type devices. This type of hand control device provides rate control of the manipulator in a cartesian coordinate frame. The RMS uses two of these devices to control its six degrees of freedom; one for X,Y, and Z and the other for roll pitch and yaw. More recently, rate control devices have been developed that allow the operator to control six degrees of freedom with one hand.

For smaller, more dextrous manipulator systems, hand control devices that provide position control instead of rate control are often preferred. A position control system provides positional correspondence between the hand controller and the slave manipulator. That is, the slave manipulator tracks the positional motions of the hand controller. This method of control is preferred for dextrous manipulation because it more closely simulates the natural hand-eye coordinated arm motions of the human operator.

There are two basic types of position hand controllers; the replica master and the so-called mini-master. The replica master is built to be a kinematic replica of the slave

manipulator. There is, therefore, a joint for joint correspondence between the motions of the master and the slave. A major advantage of such a system is that no mathematical transformations are required between the master and the slave to determine motions. The primary disadvantage of this configuration is that the master hand controller must be equal in size to the slave manipulator. This may require a prohibitively large working volume at the workstation. The mini-master reverses this situation. It is not a kinematic replica of the slave and can be built considerably smaller than the slave. However, kinematic transformations between the master and the slave are required which adds a major computational load to the control of the system.

2.2 Manipulator Attributes

Manipulator systems designed explicitly for teleoperation are generally designed to minimize weight and inertia. They provide ample compliance at the manipulator end-point through backdrivability. These features are necessary to prevent damage if the operator inadvertently bumps objects in the workspace or requires compliance during assembly and disassembly operations. Such requirements usually result in a mechanically flexible manipulator design. This tends to preclude a feature generally required for autonomous robot systems. This feature, repeatability, is the capability to return to a prerecorded manipulator pose autonomously. In fact, the capability to record poses does not even exist in most teleoperation systems. Lack of repeatability is not a problem in a teleoperated system because the operator is always visually in the loop.

2.3 Operator Feedback

Visual information from the worksite is clearly the most important feedback to the operator when performing a teleoperation task. In some cases the operator can see the worksite directly but usually he or she works from monitors in the workstation connected to video cameras at the worksite.

A second, less important but critical, source of feedback information to the operator is force and torque. That is, the operator needs to know what forces and torques are being applied at the worksite. The most sophisticated method of supplying the operator with this information is a system called bilateral force reflection. This system allows the operator to feel the slave manipulator forces and torques in the hand controller. This capability greatly enhances the operator's ability to perform delicate and/or dextrous operations. Implementation of bilateral force reflection adds considerable complexity to the teleoperation system. The hand controller must have actuators with which to impart forces and torques to the operator. This means that the hand controller is actually a robot itself and must therefore have its own robot control system. The actual

implementation of bilateral force reflection then becomes the cross coupling of the master and slave robot control systems into a single integrated control loop. Several such systems are in operation today in nuclear and undersea applications.

3.0 Robotic Systems

The manufacturing industry typically operates its manipulator systems autonomously. This is generally referred to as robotic operation. These systems must operate very rapidly and accurately in order for them to compete economically with the human labor force. This is accomplished by creating a highly structured worksite for the robotic system. This can be done fairly easily on a factory floor where equipment and work pieces can be accurately positioned and where conditions and access can be tightly controlled.

3.1 Manipulator Attributes

Manipulator systems designed for factory floor applications require good repeatability because all operations are pretaught and are performed repetitiously with no human intervention. However, human intervention is required when problems arise because the typical manufacturing robotic system has little or no sensing capability to detect anomolous conditions or to modify actions to adapt to changes in the work environment. Manufacturing robots are very massive and stiff in order to provide the needed repeatability. These manipulators also must move rapidly, and therefore, have large actuators which consume considerable amounts of power. The excessive weight and power of these manipulator systems does not generally pose a problem in the factory setting because the robots are fix mounted to the floor and are rarely transported. There is also ample power and cooling capability available on the factory floor. This is in contrast to the typical field operations of a teleoperated system where weight and power must be conserved.

3.2 Advanced Autonomous Control

There is extensive research currently underway investigating advanced autonomous control techniques for robotic systems. These techniques, as they are developed, will allow a robot to operate in environments with increasingly less imposed structure. The following capabilities are necessary for a robot to operate truly autonomously: (1) The robot must have a strategic plan. This is a high level plan of objectives, sequences of operations and recoveries in case of contingency. Such autonomous planning capabilities are of great interest to the artificial intelligence community. (2) At a lower planning level, the robot must have the ability to navigate in the workspace including manipulator path planning and collision avoidance. This will require extensive on-board geometric

reference models, machine vision and image analysis capabilities with the capability to recognize objects in ambiguous lighting. (3) The robot must have the ability to relate observations to models to confirm that a scene is consistent with expectations. (4) Each action that the robot takes will require a success confirmation criteria and alternate sequences in case of failure.

4.0 Telerobotic Systems

A telerobotic system is a hybrid of a teleoperated and robotic system. The human operator is currently required to perform the advanced autonomous control activities discussed above such as image interpretation and planning. Current robotic systems are capable of performing tasks of lesser scope autonomously when such tasks can be well defined and reasonably well structured. A telerobotic system must be designed so that the operator can do what he does best and the robot can do what it does best. As the autonomous capabilities of the system increase, the human can retract into more supervisory roles. This is the approach that is being followed in the development of the FTS.

As discussed, existing teleoperated and autonomous systems were designed to quite different sets of requirements for quite different applications. The FTS system design must encompass the capabilities of these currently existing teleoperated and robotic systems. In addition, the FTS system design must bridge a substantial gap between these two types of systems. In order for the FTS system to be telerobotic and be capable of evolution toward more autonomous operations it must have the following attributes: (1) it must support sophisticated teleoperation including features such as bilateral force reflection, (2) it must have the mechanical and computational capabilities for autonomous control, (3) it must allow easy blending of teleoperated and autonomous modes, and (4) it must have designed-in capabilities for evolution. Of particular importance here is that the system have a well structured control architecture which encompasses future needs. To fulfill this last requirement the FTS has adopted the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) [1].

5.0 FTS Design Requirements

The following paragraphs highlight some of the specific design requirements that have been imposed on the development of the FTS to ensure that it has the telerobotic capabilities discussed above. [2]

5.1 Two, 7 Degree-of-Freedom Manipulator Arms

The FTS will have two, 7 degree-of-freedom, manipulator arms. Only six degrees of freedom are required to establish the position of the end effector in space. The seventh degree of freedom allows placement of the elbow as well as the end effector providing the capability to reach around obstacles in the workspace. The seventh degree of freedom also provides additional capability for smoother control through the avoidance of manipulator singularities. Each FTS manipulator arm will have a repeatability of 0.005 inch in position and +/-0.05 degree in orientation. The required repeatability will enable the FTS to be programmed to perform simple repetitive actions autonomously.

5.2 Teleoperation

The FTS will be capable of operating with and without bilateral force reflection. The FTS will also provide the capability for resolved rate control of the manipulators and the control of individual joints.

5.3 Shared Control

The FTS will be capable of shared control in which the operator control motion in one or more coordinate axes, and the telerobot autonomously controls the motion in the other axes. This will include active compliant control. (Active compliant control means that the compliance of the robot, when it contacts the environment, is controlled by sensing forces and torques and reacting to them in an active control loop. This removes the need for the robot to be mechanically compliant in its design as required by current teleoperated systems.)

5.4 Dual-arm Coordinated Control

The FTS will be capable of dual-arm coordinated control of a grasped object using a single hand controller. This is a "semi-autonomous" capability which controls both manipulator arms in a coordinated fashion so that the operator need only concern himself with the motion of the grasped object.

5.5 Supervised Autonomous Control

The FTS will be capable of supervised autonomous execution of selected operations, such as those that are required frequently. Any autonomous task activity will be functionally assumable in real-time by teleoperation. The FTS will provide the capability for smooth and coordinated transfer between teleoperated and autonomous modes as a routine operation.

5.6 Camera and Lighting System

The FTS will have a vision subsystem that will include at least four video cameras, one on or near the wrist of each manipulator and two positionable cameras for task specific worksite viewing. The vision subsystem will also include lighting of controllable intensity adequate for task viewing. The vision subsystem will provide the capability for the remote control of positioning, orientation, zoom, focus and aperture of each camera. The capabilities of the vision subsystem are intended to support teleoperation. However, the same system is intended to support machine vision as these capabilities are added to the FTS.

5.7 Workstation and Hand Controllers

The FTS workstation will provide the capability for one person to control and monitor the telerobot in all of its teleoperated and autonomous control modes. The workstation will contain video monitors connected to the vision subsystem which allow the operator to control the telerobot without direct viewing. The workstation will contain two independent hand controllers for simultaneous control of the two telerobot manipulator arms. The workstation will provide control methods for the vision subsystem such that the operator can control the cameras and lighting while his or her hands are both engaged in manipulator teleoperation.

The FTS handcontrollers will be configured to provide the operator with easy access to the full envelope of the manipulators' range of motion. They will accommodate position control of the manipulators with and without bilateral force reflection including operation of the end effectors. The hand controllers will also be configurable to accommodate rate control of the manipulators and operation of the end effectors.

5.8 Control Architecture, Computers and Software

There are three architectures associated with the control of the FTS system; the functional architecture, the software architecture and the computer architecture.

The functional architecture, NASREM, (see reference [1]) defines the functional elements and basic structure of the system. It defines interfaces for system modularization and provides structure for interleaving of teleoperation and autonomous control. The NASREM architecture provides common reference terminology for sub-elements of the FTS control system and will provide interface definition between these sub-elements. Therefore, integration of sub-elements from different organizations and vendors is possible. This will be particularly important to the future evolution of the FTS.

The software architecture supports the functional architecture. It defines input, processing and output of each functional module. The software architecture specifies execution timing and control interdependencies. It also specifies data structures and data flow.

The computer architecture supports the functional and software architectures. The processors and inter-processor communication capabilities must be configured to provide adequate computational speed and memory capacity. After all control requirements have been satisfied for the initial implementation of the FTS it is additionally required that there be 50% CPU capacity and 35% RAM margin for future software growth and evolution.

5.9 Safety Requirements

In the space environment safety is of utmost importance. All possible precautions must be taken to prevent the FTS from injuring an EVA astronaut or from damaging a mission critical element of the shuttle or the space station. Because of this, special requirements have been placed on the safing of the FTS in case of failure.

The FTS will have the capability to detect failures within its subsystems and operations and to automatically assume a safe state upon such detection. This capability will be two-fault tolerant which means that the detection and safing hardware and software itself must be capable of sustaining two internal failures and still performing the necessary actions.

6.0 Conclusions

The technology base for FTS development will come, primarily, from manipulator systems developed for nuclear and undersea applications and from robotic technology developed for the manufacturing environment.

A system which effectively combines both teleoperation and autonomous capabilities has never been built. Additionally, the FTS must have the designed-in capability to accommodate more advanced technology elements as they become available. To accomplish these goals, the FTS design must incorporate sophisticated sensor-driven control techniques into a well structured control architecture which supports good teleoperability as well as the necessary attributes which allow the system to be programmed for autonomous operations. No new basic technology elements need to be developed to enable the development of the FTS initial capabilities. However, the

design, integration and operation of a telerobotic system of this complexity will represent an important advance in the state of the art in the field of robotics.

7.0 References

[1] J. Albus, H. McCain, and R. Lumia, "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," SS-GSFC-0027, December 1986.

[2] "Flight Telerobotic Requirements Document for Phase C/D", SS-GSFC-0043, Attachment B.

THE FLIGHT TELEROBOTIC SERVICER TINMAN CONCEPT: SYSTEM DESIGN DRIVERS AND TASK ANALYSIS

J.F. Andary, D.R. Hewitt, S.W. Hinkal

NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

During 1988, the National Aeronautics and Space Administration (NASA) conducted a 9-month in-house Phase B study to develop a preliminary definition of the Flight Telerobotic Servicer (FTS) that could be used to understand the operational concepts and scenarios for the FTS. Called the "Tinman," this design concept was also used to begin the process of establishing resources and interfaces for the FTS on Space Station Freedom, the National Space Transportation System shuttle orbiter, and the Orbital Maneuvering Vehicle.

Starting with an analysis of the requirements and task capabilities as stated in the Phase B study requirements document, the study identified eight major design drivers for the FTS. This paper will describe each of these design drivers and their impacts on the Tinman design concept.

Next, this paper will discuss the planning that is currently underway for providing resources for the FTS on Space Station Freedom, including up to 2000 W of peak power, up to four color video channels, and command and data rates up to 500 kbps between the telerobot and the control station.

Finally, an example will be presented to show how the Tinman design concept was used to analyze task scenarios and explore the operational capabilities of the FTS. A structured methodology using a standard terminology consistent with the NASA/National Bureau of Standards Standard Reference Model for Telerobot Control System Architecture (NASREM) was developed for this analysis.

1. Introduction

The Flight Telerobotic Servicer (FTS) will be used on the National Space Transportation System (NSTS) shuttle orbiter and Space Station Freedom to assist the astronauts in performing assembly, maintenance, servicing, and inspection tasks. Although it is primarily a teleoperated device at first, the FTS is being designed to grow and evolve to higher states of autonomy. Eventually, it will be capable of working from the Orbital Maneuvering Vehicle (OMV) to service free-flying spacecraft at great distances from the space station. A version of the FTS could also be resident on the large space platforms that are part of the space station program.

This paper discusses the technical design drivers that the in-house Phase B study identified as significant in the development of such a robotic system for space. The Phase B study started with the initial requirements of the top-level mission, system, and functional requirements for the FTS [1]. These requirements were developed during a Phase A study conducted by NASA during the fall of 1986 [2 and 3].

The output of the in-house Phase B study was integrated with the results of more in-depth Phase B studies conducted by Martin Marietta Astronautics Group in Denver, CO, and Grumman Space Systems in Bethpage, NY, to generate the requirements for Phases C and D of the FTS Program, which are expected to begin in the Summer of 1989 [4].

2. Study Approach

The Phase B study [5] started with a detailed analysis of the space station tasks described in the requirements document [1]. These tasks describe generic capabilities that are intended to be representative of the fundamental mission of the FTS as a robotic device that assists the astronauts in assembly, maintenance, servicing, and inspection tasks in the unpressurized environment of the space station.

Analyzing the tasks in the requirements document [1] led to the identification of a number of design drivers for the development of the FTS. These design drivers resulted in a series of trade studies that were used to develop candidate design solutions. The resulting design concept for the FTS was called the "Tinman." This concept resulted in a robotic system that was adequate for the assigned tasks and could perform the tasks reliably and safely.

Advanced technology items were scrutinized as to their relevance to the performance of the assigned tasks, as well as to their state of readiness. If an item was not considered necessary, it was not incorporated into the design. Some items were considered appropriate, but their state of readiness made them too high a risk for inclusion in the initial implementation of the FTS. High technology should not be used just for the sake of using it, if it should then fail in orbit. An early failure of the FTS would be a great setback for space robotics. Instead of serving as a useful tool for the astronauts, the FTS would be discarded, and the astronauts would turn to another means of accomplishing the tasks.

A program requirement is that the FTS must be capable of growth and evolution. System adaptability is necessary because of the emerging technologies that will be valuable to the program once they have matured. The FTS must be designed from the ground up with the proper "hooks" and "scars" for growth. With the appropriate systems engineering and architectures that can accommodate growth, advanced technology with software and hardware can be added later to the system with minimum impact. To accomplish this, NASA has adopted a control architecture developed by the National Institute for Standards and Technology (NIST), formerly the National Bureau of Standards (NBS), that permits this type of growth [6].

3. Design Concept

Figure 1 shows the design concept that was developed for the FTS during the Phase B study. As shown in the drawing, the telerobot is composed of three major subassemblies: the main body, the manipulator arm assembly, and the arm-positioning system.

The main body contains all the major electronic components of the telerobot, as well as the grapple fixture by which the telerobot is picked up by one of the large manipulator arms (e.g., the Space Station Remote Manipulator System (SSRMS) or the NSTS Remote Manipulator System (RMS)). The main body also contains the attachment grapple (or foot) by which the telerobot is securely fixed at the worksite.

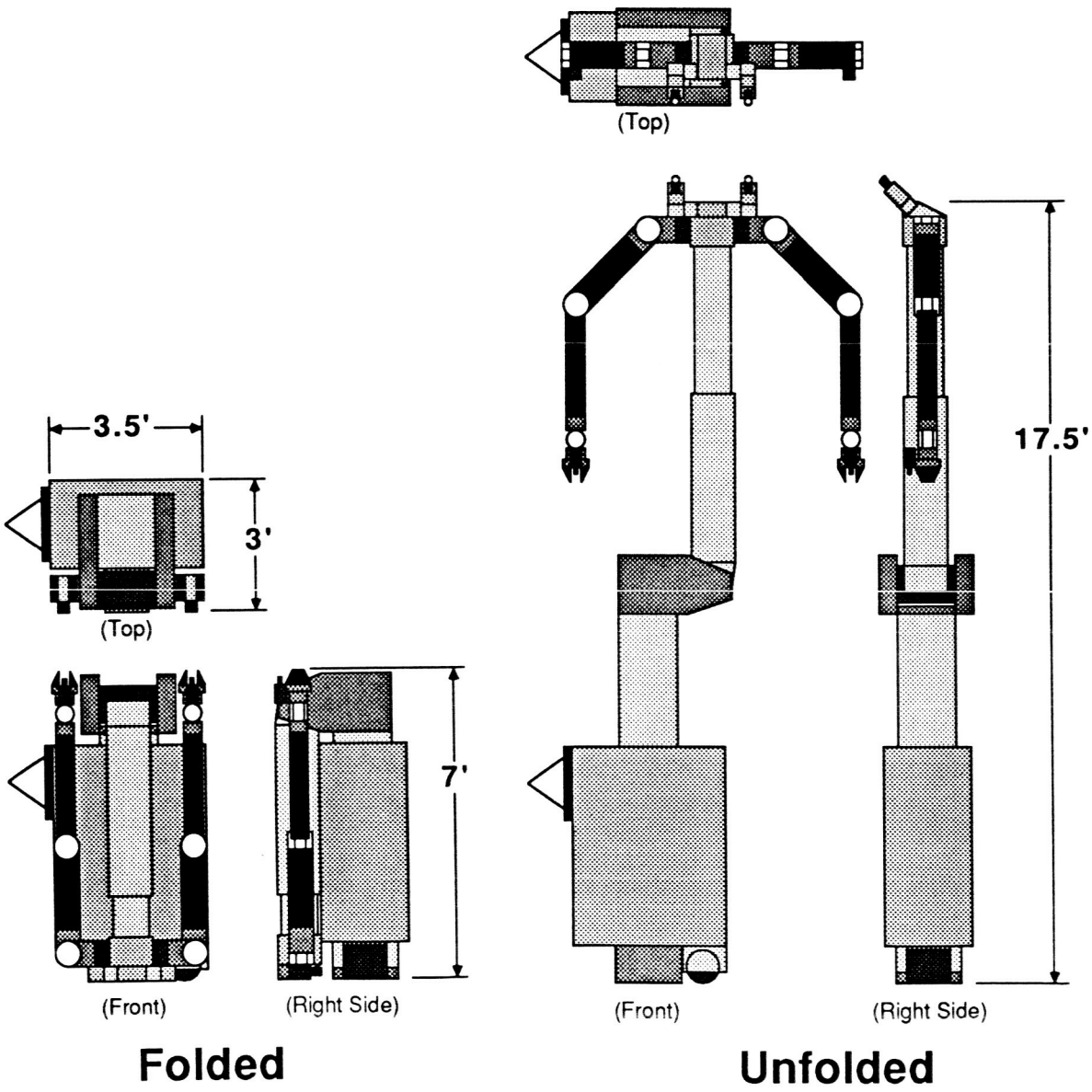


Figure 1. Flight Telerobotic Servicer (FTS) Dimensions

One of the features of the main body of the telerobot is that it is free to rotate about its central core and the attachment foot. This freedom to rotate allows the thermal radiators, which cover three sides of the main body, to be oriented for optimum heat rejection at the worksite. Main body rotation with respect to the attachment foot allows the operator of the large manipulator arm (SSRMS or RMS) another degree of freedom to help orient the FTS foot for proper mating to the worksite attachment point.

The next major component of the telerobot is the arm positioning system that consists of two, linearly driven, tubular sections connected through an offset rotational joint. The lower section is free to rotate simultaneously with respect to both the main body and the attachment foot. The manipulator arms are free to rotate $\pm 180°$ with respect to the upper section. Five degrees of freedom are obtained to position the arms relative to the telerobot main body and attachment location. There are a number of advantages to the arm positioning system: it extends the reach of the telerobot without extending the length of the manipulator arms; it allows the arms to be positioned squarely to a task so that the teleoperator approaches the task in a natural manner; and it allows the telerobot to reach out over large objects that may come between the attachment fixture and the location of the task.

The final component of the telerobot is the manipulator arm assembly that is mounted to the end of the positioning system. It consists of the shoulder assembly that rotates $\pm 180°$ about the end of the positioning system, and two seven-degree-of-freedom manipulators mounted to each end of the shoulder assembly. The manipulators are 1.524 m (60 in.) long and are configured with a roll-pitch-roll shoulder, pitch in the elbow, and roll-pitch-roll in the wrist.

In addition to the telerobot, the FTS includes two workstation designs: a stowable workstation for the NSTS, which will be mounted in the aft flight deck of the shuttle and the space station workstation, which will include FTS-unique hardware that will be incorporated into the space station Multipurpose Application Console (MPAC).

4. Design Drivers

During the analysis of the requirements and the task capabilities, the study team identified the following major design drivers for the FTS:

- Thermal environment
- Independent operation
- Manipulator stability and positioning
- Safety
- Mobility
- Evolution
- 1-g operation
- Human interface

The impact of each of these design drivers on the final design concept will be discussed in the following paragraphs. Not all of the design drivers are independent. Often, more than one of the drivers affects the design of a particular subsystem. Therefore a systems approach had to be taken to the trade studies to determine the appropriate solution leading to the best overall design concept.

4.1 Thermal Environment

The thermal environment created by the vacuum of space introduces unique problems for the FTS in an area that is only a minor concern for terrestrial robots. In space, the only way of dissipating heat is by radiation or conduction. The only paths for conduction were by hookup to the space station thermal system or by dumping heat into the FTS base mounting structure. Both options were considered too restrictive for the flexibility and usefulness of the FTS and they also created a thermal interface to the space station that the design team wanted to avoid. Therefore, radiation is the only acceptable means of heat dissipation.

Dissipating heat from a robot with peak operating power in the 1- to 2-kW range with approximately 20 motors, several high-speed computers, video equipment, and batteries with radiation as the only means of cooling is a thermal problem. The operation of the FTS should not be restricted because of the thermal environment. This meant that the FTS had to be capable of operating with arbitrary Sun angles and with partial blockages from the structure at the worksite.

To overcome these problems, the overall power of the telerobot was reduced, its total radiating capability was increased, and the main battery was removed from the telerobot.

One effect of reducing the power was the selection of motors at each joint that were sized for the tasks in zero gravity but could not operate without assistance on Earth. By using smaller motors, the manipulator thermal system could be separated from the rest of the body, and all the other heat-dissipating components could be collected into one structure that could be optimized for thermal radiation.

Figure 2 shows the concept for the telerobot body that uses heat pipes to direct the heat from the electronic boxes out to the outside surfaces, where radiators cover three sides. The main body was designed to rotate independently of the manipulators and the arm-positioning system, so that it could be controlled to track an optimal orientation to cold space as the telerobot performs its tasks.

Removing the main battery from the telerobot had a number of effects on the design. It reduced the mass of the telerobot and removed a source of power dissipation. It also freed the telerobot from the tight thermal limits that the battery imposed on the system.

The combined effect of all these design choices produced a thermal design that is independent of the space station and that will permit indefinite operation of FTS under most conditions. In some extreme cases of radiator blockage, the task may have to be halted temporarily to allow the telerobot to cool down. Consideration was made of the use of a small "backpack" composed of Phase Change Material (PCM), which could be used to absorb peak loads to enable the telerobot to continue operating for a brief time under extreme conditions. The thermal system is also an ideal candidate for the incorporation of an expert system that could continually monitor the thermal health of the telerobot and inform the operator of the amount of time that is left before a cooldown period would be required.

4.2 Independent Operation

Another requirement is that the FTS must be capable of limited operation independently of hard-wired utilities for power, data, and video from the space station.

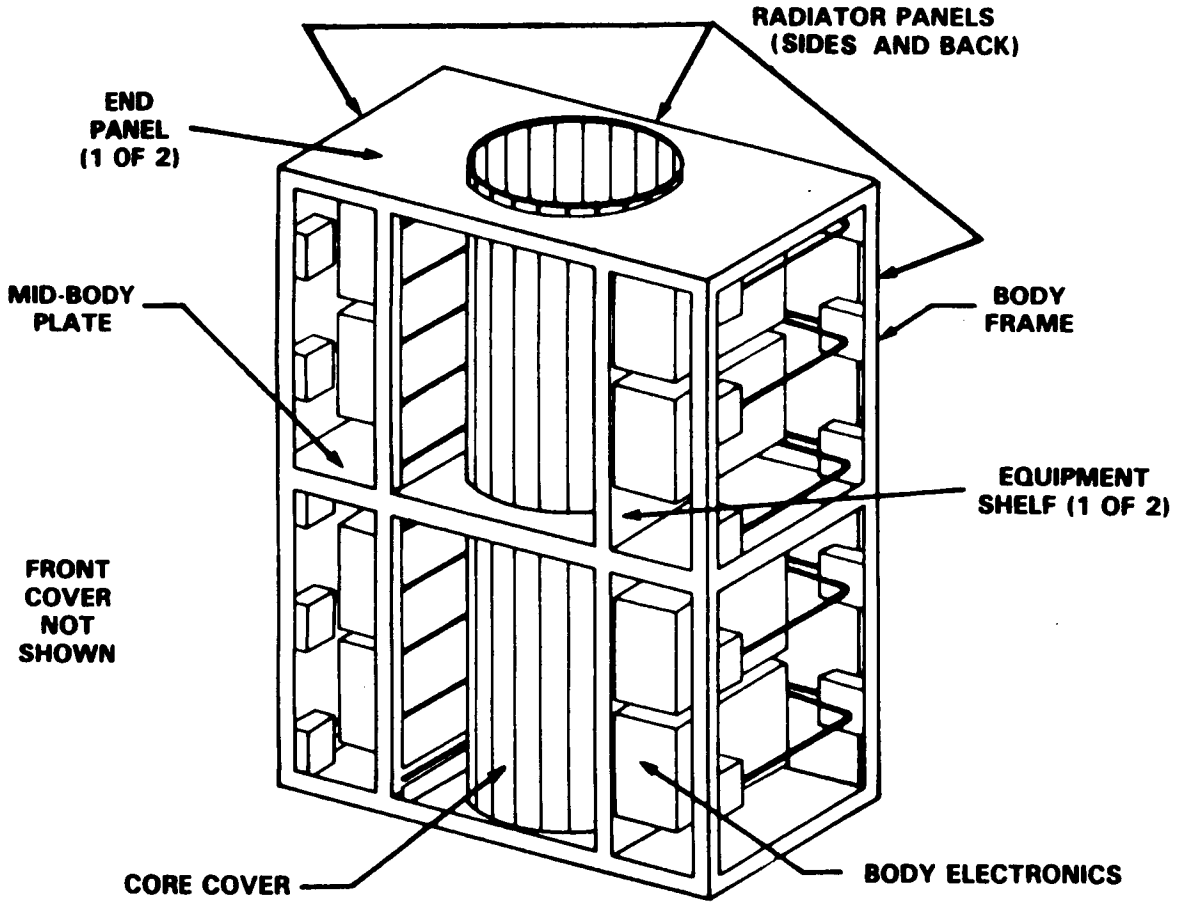


Figure 2. Structure Subsystem Tinman Design

Because of this requirement, a large battery and a radio frequency (RF) communications system were included in the design of the FTS. The FTS can never be totally independent of the space station, because it always needs a firm structural attachment when working. However, the requirement for independent operation gives the FTS a tremendous amount of flexibility, allowing it to work in areas on the space station where no utility ports are located.

A battery that would allow operation for even a few hours at the power levels of the FTS adds considerable weight and adversely impacts the thermal subsystem. Since the independent operation is not the primary mode of operation, it was decided to remove the battery and the communications system from the main body of the telerobot and locate them in a separate module called the Robot Support Module (RSM). Because there is not a requirement for an early independent operational capability, the RSM could be launched later than the FTS, thereby reducing the initial manifested weight of the FTS.

Another advantage of the separate RSM is that it would be possible to design different RSMs for the different operating environments of the FTS. Because the NSTS and the space station have different power and communications systems, a different RSM could be designed for each location. Another RSM could be built for operation from the OMV for the servicing of free-flying spacecraft away from the NSTS orbiter or the station, as shown in Figure 3. Two RSMs on the space station itself are a possibility, so that while one is being used, the other could be having its battery recharged.

4.3 Manipulator Positioning and Stability

When the work environment of the FTS is examined in both the shuttle payload bay and on the space station, the same dimension of 5 m keeps reoccurring. The shuttle payload bay is 4.57 m (15 ft) wide, and, consequently, most payloads launched by the shuttle are also approximately 5 m wide or 5 m in diameter. The space station truss bays are 5-m cubes and the Attached Payload Accommodation Equipment (APAE) sit on a 5- by 5-m base. It can be concluded from this information that the ideal reach envelope of the telerobot would be 5 m. If the telerobot is to work in these locations, it must be able to cover these types of distances. However, early analysis indicated that a 5-m reach for the manipulator arms was not feasible if the telerobot was to do any dexterous manipulation. A local mobility system and an arm-positioning system were chosen to deliver the arms to the task. This approach allows the arms to be shorter and more rigid for the fine control tasks.

Figure 4 shows the reach envelope of the telerobot. Situated in the center of the space station truss bay, the telerobot can reach all faces of the bay. The reach of the telerobot at an APAE site is shown in Figure 5, where the Orbital Replaceable Units (ORUs) in the center can be reached from either side, even if larger ORUs are in the way.

The flexibility and controllability of such a system are still areas of concern that are being investigated. Preliminary indications are that the arm-positioning system can be made rigid enough to meet the task requirements. The five degrees of freedom in the arm-positioning system are controlled open loop and, therefore, do not contribute complexity to the arm control problem. The degrees of freedom in the positioning system are commanded to set positions one at a time and are then rigidly locked before the operator begins to use the manipulator arms. It is not anticipated that the

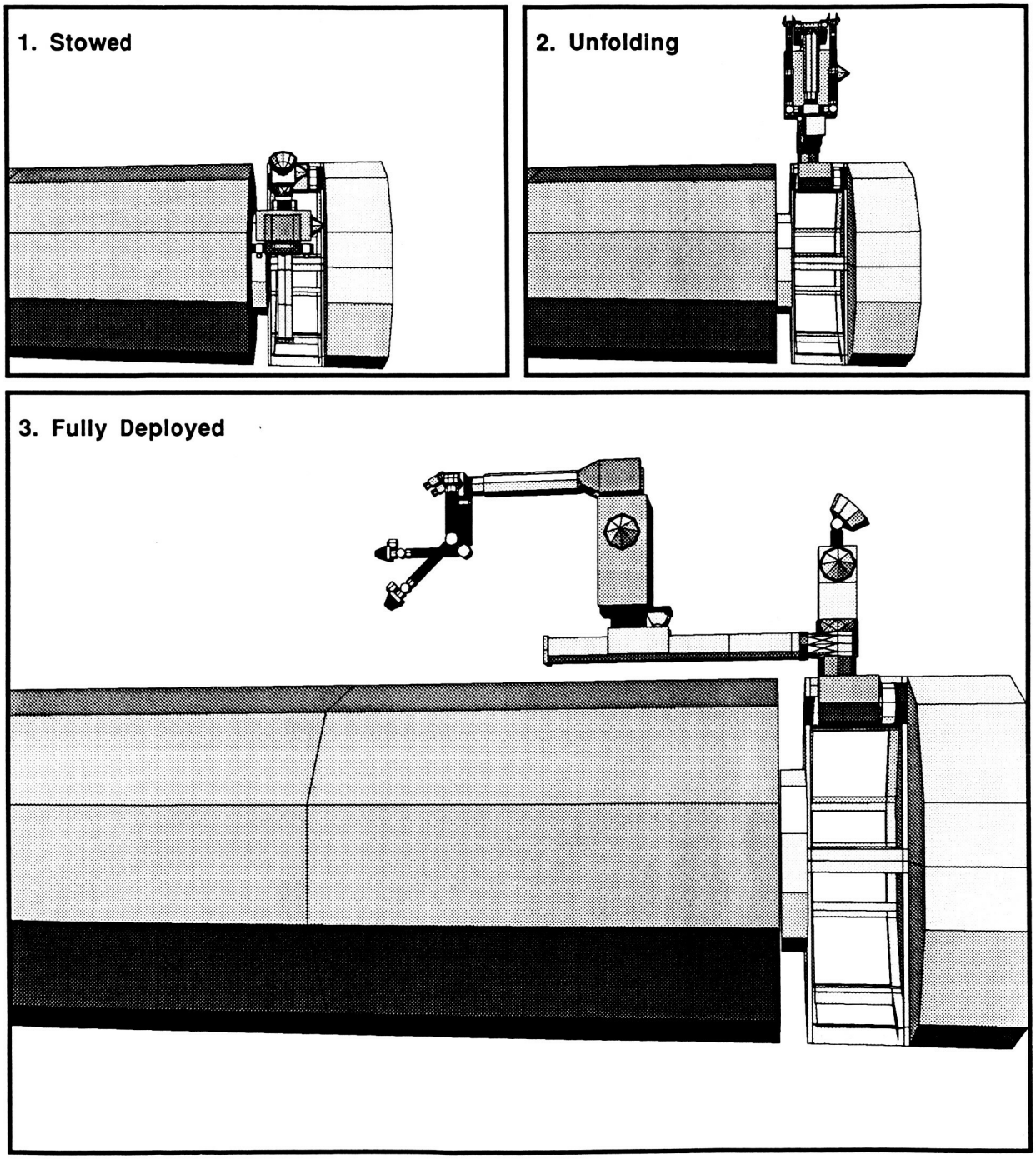


Figure 3. FTS/Orbiting Maneuvering Vehicle Servicing

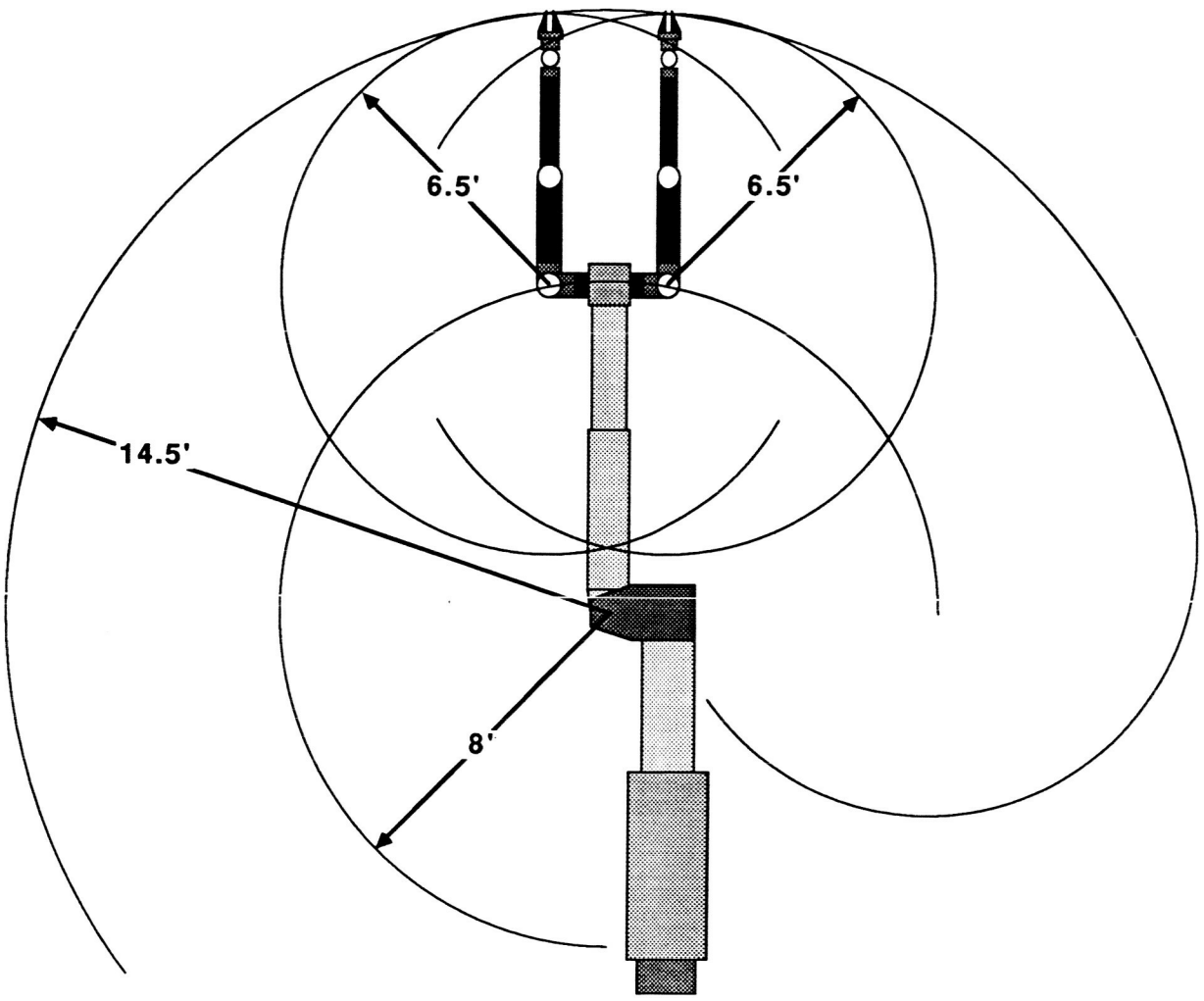


Figure 4. FTS Work Volume

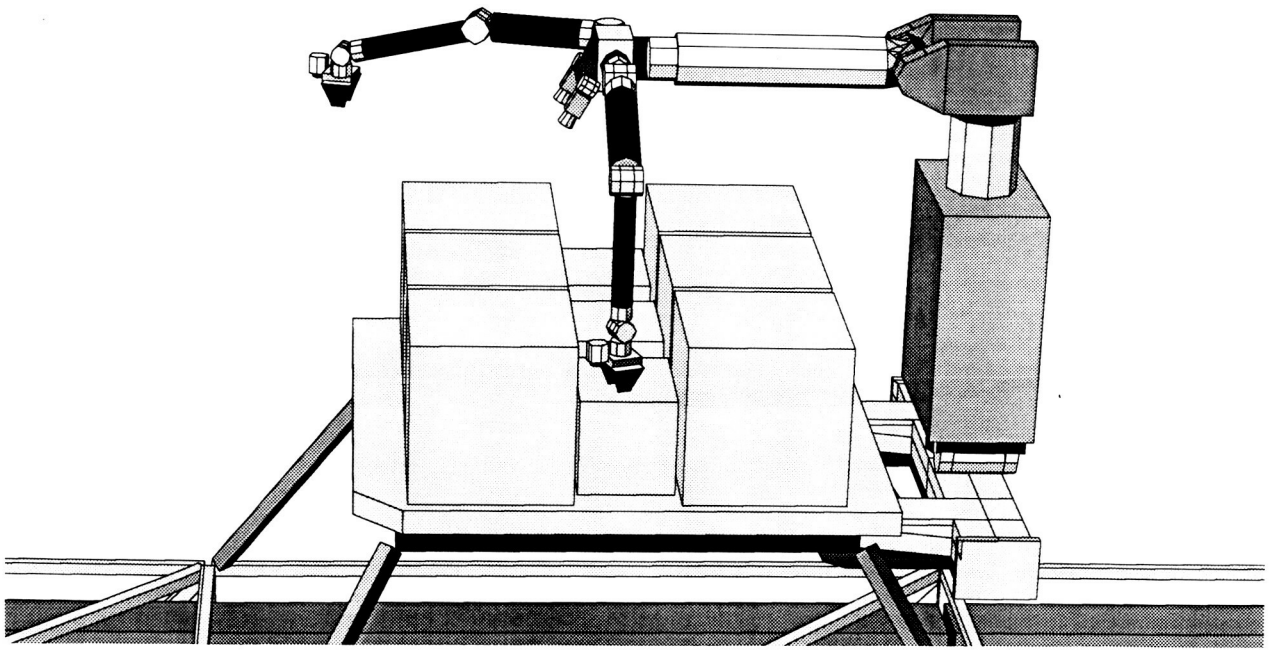


Figure 5. FTS Operating From Attached Payload Accommodation Equipment

positioning system would be teleoperated through the hand controllers. The operator could simply key in the position of the joints from a keyboard.

4.4 Safety

Safety is of primary importance in the design of the FTS. Safety influences each subsystem and must be designed into the FTS from the start. The Phase B study approach was to set up a watchdog safety subsystem that consists of redundant radiation-hardened computers and associated sensors in the telerobot to monitor all aspects of the telerobot operations and health. Also, the workstation has a safety computer that acts as a global safety monitor for workstation operations, as well as the telerobot safety subsystem. Whenever any anomalous condition is detected, the safety computers will stop all movement of the telerobot.

There is also a safety shutdown signal that originates from an astronaut on Extravehicular Activity (EVA) if he senses a problem with the telerobot. This is called the EVA safety link and allows an astronaut on EVA to have shutdown control of the telerobot whenever he works in the vicinity of the telerobot.

Each controller for the manipulator joints is capable of being programmed to limit the local parameters associated with that joint, such as velocity, acceleration. This programming allows the motions of the telerobot to be tailored to the task and the environment. A velocity limit of 0.30 m/s (1 ft/s) is imposed on the manipulators whenever the telerobot is working in the vicinity of an astronaut or critical hardware. Similar limits must be imposed on the maximum momentum the system can attain when moving an object. This may result in an even lower tip velocity, but it ensures that the telerobot can safely brake its motion to avoid collision.

Another safety feature in the telerobot is the inclusion of a small, holdup battery within the telerobot to sustain its functions and to perform an orderly shutdown in the event of a power loss. This safety feature is needed when the telerobot operates without the large battery in the RSM, and derives its power from the host vehicle.

4.5 Mobility

Mobility was identified early as an FTS design driver. There is not a requirement for the type of mobility that would allow the telerobot to walk down the space station truss. There are other means available on the shuttle and the space station to provide global mobility, such as the RMS on the shuttle and the SSRMS on space station attached to a transport device such as the Mobile Servicing Center (MSC) or the Mobile Transporter (MT). However, from a close examination of the FTS tasks, it is clear that some form of "local mobility" (or "robility") was needed at the worksite in order to make the FTS a useful tool on the space station.

The local mobility system that is part of the in-house concept is a portable rail that can ride out to the worksite with the telerobot to provide lateral movement. The portable rail, together with the arm-positioning system, allows the manipulator arms to be positioned with six degrees of freedom at the worksite. The length of the portable rail had to be traded off against the flexibility of the rail and the induced motions at the end of the rail that occur when the telerobot is in operation. The portable rail is attached to the RSM in the in-house concept, so that the telerobot/rail/RSM combination can be picked up as one unit and carried to the worksite by one of the transport devices on space station. Figure 6 shows the portable rail supporting the telerobot from the RSM.

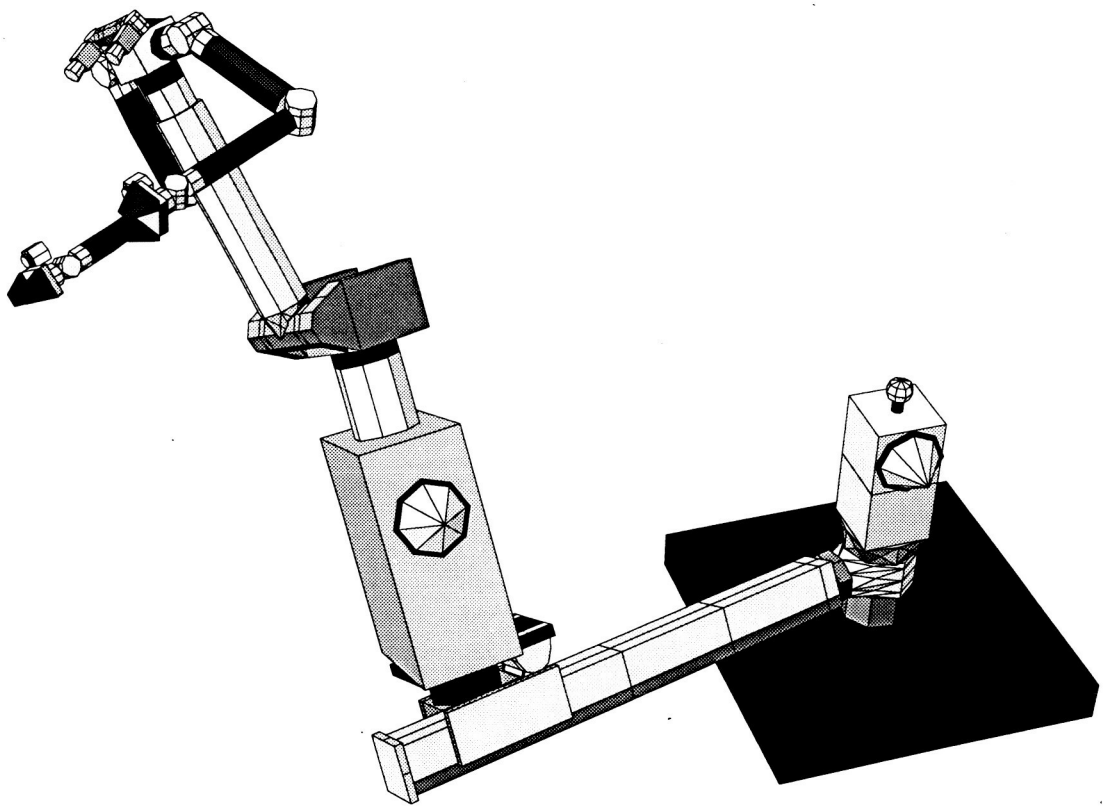


Figure 6. FTS and Robot Support Module

4.6 Evolution

The FTS must be able to evolve towards greater adaptability that includes more autonomous operation accomplished through the incorporation of advanced hardware and software items as they become available. Since the FTS is intended for permanent residence on the space station, new items must be added to the system in orbit. The FTS must be designed to easily accept these changes. This will be done by the incorporation of modularity and accessibility in the design of all subsystems of the FTS and by a careful implementation of the NASREM architecture.

Primary growth areas are expected to be in more advanced computers, upgraded software, advanced sensors with image processing, smart end effectors, and new and more efficient power systems. Also, the manipulator arms could be of a modular design, so that they can be reconfigured to provide more capability for new maintenance and servicing tasks on the space station. Power, data, and video lines would run throughout the telerobot with standard interfaces defined at the tool plate, arm joints, and other locations where hardware may be added or later changed.

A vision system, which initially is just a closed circuit video system, can easily grow to a stereovision system and eventually evolve to full machine vision. Steps that can be taken in the initial design to facilitate this growth are the choice and location of cameras and the interfaces to permit the computers to have access to image data.

4.7 1-g Operation

Requiring that the telerobot exhibit its full operational capability in the gravity environment of Earth has far-reaching impact on the system design. From a programmatic standpoint, the FTS must be capable of being tested in the performance of representative tasks on Earth before it is committed to launch. However, such a requirement has to be weighed against the impacts it causes on the structural, controls, electromechanical, power, and thermal subsystems.

For terrestrial robots, a 100:1 weight-to-lift ratio is not unusual, and a ratio of 10:1 is just now being achieved by some research manipulators, such as the Laboratory Telerobotic Manipulators (LTMs) being developed by NASA Langley Research Center and the Oak Ridge National Laboratory. This means that if the FTS were required to handle mockup hardware weighing 50 lb, the manipulators would be on the order of 300 to 500 lb each when using today's technology. This results in 600 to 1000 lb for just the manipulators. The total manifested weight for the FTS, including the telerobot and the workstation, is currently 1500 lb.

The FTS must undergo a strict weight control program that will result in motors and a structure that will be adequate to accelerate the inertias required by the tasks in the zero-gravity environment of space, but may not be capable of lifting the mockups of the same hardware on Earth. This will mean that the telerobot will need special assistance to perform its operations in 1-g, such as counterweights and other gravity off-loading devices.

Smaller, lightweight motors are a benefit to both the power and thermal subsystems of the FTS. A lighter weight structure has an impact on the control system, since the manipulators will be more flexible, but this is not viewed as an insurmountable problem for the FTS, because of the recent advances in algorithms for the control of flexible robots.

4.8 Human Interface

The design of the FTS for the human operator extends beyond the obvious human engineering of the workstation, (e.g., ensuring that the operator is presented with all the necessary displays and controls). The FTS is a teleoperated device for which the operator is directly in the control loop. The human interface has a strong influence on the design of the control system, the data system, and the sensors, including the vision system.

The FTS must be designed for operation by one operator. Inventive means must be found for the control of the cameras, illumination, and other peripheral devices when the operator uses both hands to operate the manipulators.

The study team concluded that the use of force-reflecting hand controllers should be a requirement for the FTS. This would permit the operator to sense the manipulator forces in his hand controllers. For a teleoperated device, this requirement is a tremendous asset to the operator. It enhances safety when working in an unstructured environment, and it has been proven through documented experiments in the laboratory to reduce errors and overall training time.

The problem on force reflection is the stringent data latency requirement it places on the data system for communications between the workstation and the telerobot. Because the force loop is now closed through the workstation, the stability of the control loop depends on minimizing the delay time for the round trip signal. The loop should operate at approximately 200 Hz, which results in a latency requirement of 5 ms. The FTS will use the Data Management System (DMS) on the space station to connect the workstation to the telerobot, and an assessment has to be made to see if the DMS can satisfy such a latency requirement.

5. Resource Accommodations

Just as plans to utilize humans in the space environment requires accommodations for communication and other resources, plans to use an extravehicular space robotic system requires integration of robotic interfaces and resource accommodation into that environment. Because the application of the FTS is to be in the extravehicular environment, and each task planned for execution by the FTS must be planned such that suitably equipped astronauts also can perform the task, much of the planning for the integration of the FTS can be combined with requirements for operations involving EVA. The design criteria for robot compatible hardware [7] on space stations should highlight similarities to human integration standards [8].

The requirements for robot access and interface are expected to be similar but not identical to those for the astronaut performing EVA. For example, robotic manipulation may be simplified if handling interfaces have flat surfaces of certain dimensions. An astronaut's glove may not require flat surfaces, but emphasis may be placed on rounded interfaces to prevent glove damage. Only by studying both sets of requirements can both be met and a solution to both be agreed upon. In this case, both needs can be met with handles designed with flat surfaces and rounded edges. However, even this simple case is not yet resolved, because the FTS and its tools and end effector designs have not yet been chosen, and the designers of space station hardware have only begun their interface designs.

The process of integrating the FTS into the space station is ensured because the space station program office has directed that, for extravehicular operations, all

hardware be designed for telerobotic manipulation as the baseline with backup by astronauts performing EVA when practical and cost effective.

5.1 Control Station

The Space Station Freedom Program Definition and Requirements Document [9] currently identifies locations for the FTS control station within the pressurized resource nodes that connect the larger pressurized modules and house the primary station control functions. Each FTS control station will consist of the space station common workstation, augmented with unique items such as hand controllers, software, and any unique safety controls that may be required, such as for rapid shutdown of telerobotic operation.

5.2 Transportation

The space station MSC will provide transportation of the telerobot from the FTS external storage location to worksites on the truss. The MSC also will support operations from the base of its Mobile Remote Servicer (MRS) and from the end of its manipulator system, the SSRMS, as shown in Figure 7. Interfaces for structural attachment, power, data, and video must be established between the FTS and the MSC to support these operations.

5.3 Access to Utilities

At each worksite, the FTS will require structural attachment points, as well as access to power, data, and video links that are consistent with the operations planned at each worksite. Where the full capability of the FTS is required, a hardline connection between the telerobot and the control station will be provided. This connection will provide a real-time bidirectional data link between the telerobot and the control station that is capable of transmitting 0.5 Mbps, with a one-way data latency not greater than 5 ms. The connection will also accommodate simultaneous transmission of four channels of color video to the control station. This will permit bilateral force reflective feedback control and the capability to display up to four views of the worksite environment from telerobot-mounted cameras.

If operations do not require the full capability of the FTS, or if a task develops at a location where hardline utilities are not available, the telerobot may be operated in its independent mode. This will allow it to operate continuously for at least 2 hours on internal battery power. Communications are established by means of the FTS wireless RF communications equipment.

Force reflection may be impossible or degraded in this mode, because RF transmissions on the station cannot accommodate the rapid communication between telerobot and workstation that is necessary for good bilateral force reflection. However, automated control and non-force-reflecting teleoperated control will still be possible and will bring a significant remote dexterous capability to the astronauts. RF operations on the space station are currently limited to the simultaneous transmission of only three video channels, but video switching and compression techniques can still provide access to all four of the telerobotic camera views at the workstation.

5.4 Worksite Locations

Studies are underway to identify the potential worksite locations for the FTS. This effort is being coordinated with the providers of the pallets which accommodate the

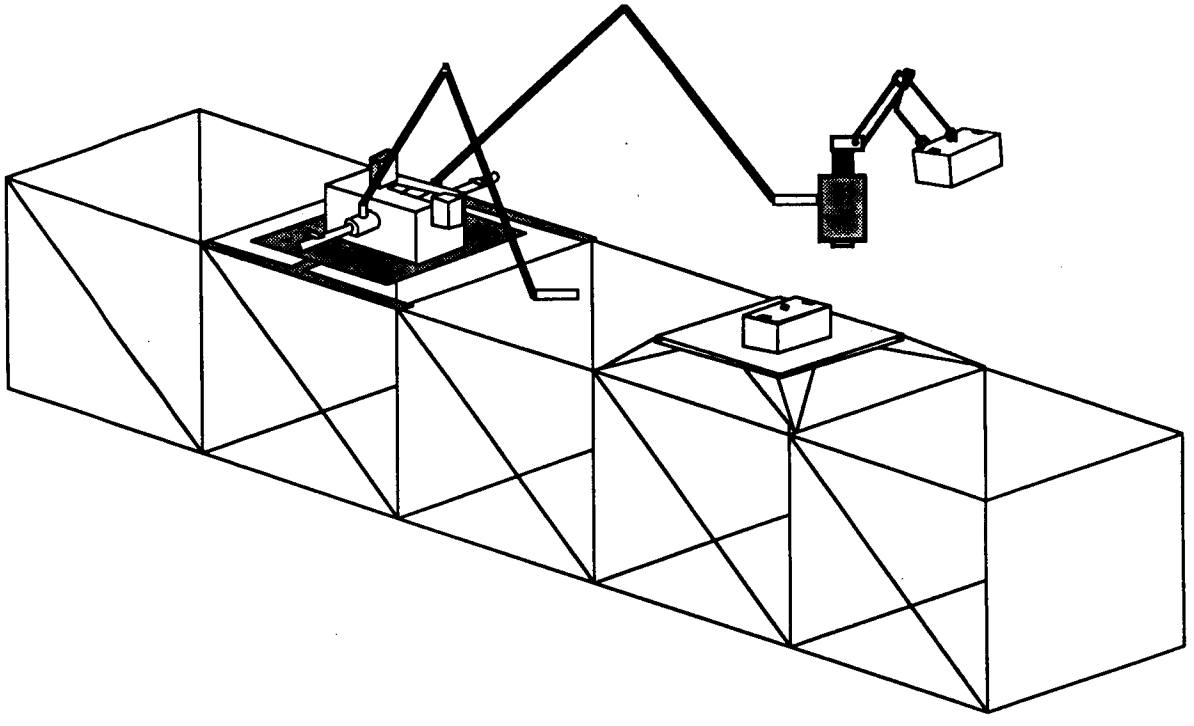


Figure 7. Transporter Attached Operation

distributed systems equipment and APAE from Johnson Space Center, Goddard Space Flight Center (GSFC), and Lewis Research Center. There are 11 locations currently identified in the space station documentation where utility ports are to be provided to support FTS operations [10]. Additional worksite accommodations will be provided to support FTS transfer of equipment to and from the Unpressurized Logistics Carrier (ULC), which is being built by Marshall Space Flight Center.

These give the FTS the capability to operate at full capability at many locations and at reduced capability virtually anywhere on the space station. Use of the FTS throughout the station is being pursued further through commonality with utility access ports for the MSC and other equipment, and by seeking distributed access to utilities where such access makes sense in the integrated planning of operations and maintenance of the space station.

6. Task Analysis

A structured task analysis methodology has been developed by the Mission Utilization Team (MUT) at the GSFC in order to analyze the FTS Tinman concept in the performance of Space Station Freedom assembly tasks. The procedures that were developed are compatible with the NASA/NBS Standard Reference Model (NASREM) [6]. The process starts at level 6 of the NASREM hierarchy with the decomposition of a mission into a sequence of tasks. On level 5, the tasks are decomposed into subtasks called steps. On level 4, these steps are then further decomposed into elementary moves (E-moves). NASREM levels 3 through 1 were not addressed in this initial effort.

Following each level of decomposition, a script was produced in which each work system, such as the RMS, the FTS, or the MSC were allocated assignments based on their capabilities. Several of the scripts were modeled with Computer-Aided Design (CAD) software to check the compatibility of the work systems, accessibility, and collision avoidance.

Finally, an assessment of the Space Station Freedom assembly tasks was made by rating system level attributes for FTS utilization as compared to EVA.

6.1 Methodology

The following account of the methodology used for this study is taken from the Flight Telerobotic Servicer Task Analysis Document [11]. The flow chart of the task analysis process is shown in Figure 8. The method consists of the main procedures described in the following paragraphs.

6.1.1 Statement of Objective

The statement of objective is the initial statement of the activity or work to be performed. It is described by the mission, task, step, or E-move. Most FTS tasks can be derived from two overall Freedom missions: assemble Space Station Freedom and maintain (or service) Space Station Freedom. For this task analysis, the objective was assembly of the Space Station Freedom elements on flights MB-2 through MB-4.

6.1.2 Decomposition

Decomposition is the process of breaking down a high-level objective, such as a mission or task, into the lower level activities and actions required to accomplish the desired objective. The activities and actions are expressed with selected verbs. An

Figure 8. Task Analysis Flowchart

example of the verbs used is shown in Table 1. Decomposition at the upper NASREM levels 6 and 5 are task or object dependent and robot independent. Missions and tasks at these levels are defined and stated in terms of actions on objects. Because of the task dependence, these levels require primarily geometric information regarding the task.

Levels 3 through 1, on the other hand, process information that is task independent and robot dependent. At these levels, the upper level geometric description is converted into a low-level robot-dependent description with emphasis placed on joint motions, work system dynamics, efficient collision-free paths, and motor servoing.

Decomposition to level 3 was done only for the six baseline tasks listed in the FTS Phase C/D Requirements Document [4] and is documented in volume 2 of the FTS Task Analysis Document [11]. It was necessary to establish the scripts and interface concepts required to perform these fundamental operations to ensure that the upper level Space Station Freedom assembly tasks could be considered composed of strings of these elementary operations.

6.1.3 Tasks

The result of mission decomposition is a sequential list of task statements. For this study, the task statements came from two sources: The Space Station Assembly Operations, Functional Flows and Resource Allocations, Flights MB-2 through MB-5 [12], and the baseline task set from the FTS Phase C/D Requirements Document [4].

6.1.4 Task Steps

Steps are characterized as actions capable of being performed by a single work system (FTS, SSRMS, etc.) and are described by providing the endpoints (initial and final location or configurations) of objects. The majority of steps tend to describe actions such as positioning, attaching, and detaching objects. An example of several steps in the Power Management and Distribution (PMAD) pallet installation is:

- o Unstow PMAD pallet from payload bay
- o Position pallet to the vicinity of starboard bay No. 1 (SB-1)
- o Deploy starboard legs
- o Attach starboard legs to the starboard nodes

6.1.5 Work System Capabilities

An essential part of the task analysis process is to gather knowledge concerning the available work systems for a task. The work systems chosen for a task will depend on both the requirements of the task and the skills or performance capabilities of the work systems. The significant requirements and performance capabilities impelled on or by work systems are the following:

- o Space Station Freedom performance limitations on work systems--maximum values of the following:
  - -- Velocities
  - -- Accelerations
  - -- Forces
  - -- Moments
  - -- Torques

Table 1
Examples of Selected Verbs in Decomposition

NASREM Level	Term Type	Explanation	Term
6	Missions	These verbs are inputs to level 6 and are decomposed into tasks.	Assemble Maintain (Freedom's hardware) Service (customer hardware)
5	Tasks	These verbs are inputs to level 5 and are decomposed into steps, which are then assigned to various work systems in a script.	Construct Dismantle Inspect Install Remove Repair Replace (changeout) Replenish
4	Steps	These verbs are inputs to level 4 and are decomposed into elementary moves, which are subsystem level commands.	Actuate Attach/detach Deploy Locate Measure Observe Position Stow/unstow
3	Elementary moves (E-moves)	These verbs are inputs to level 3 and are the subsystem level commands.	TBD

- o Work system requirements, skills, and performance capabilities--maximum values of the following:

  -- Forces
  -- Torques
  -- Moments
  -- Stability
  -- Mobility
  -- Reach
  -- Dexterity
  -- Speed
  -- Control sophistication
  -- Resource requirements

6.1.6 Scripting

Scripting is the process of developing an operational scenario for accomplishing a task with specific work systems. A script is basically a set of lines or actions (tasks, steps, E-moves, etc.), that are to be performed by a set of actors (FTS, SSRMS, etc.). The scenario reflects the capabilities and limitations of the work systems.

Scripting can occur at the mission, task, or step level. An example from the scripting of the task of installation of the PMAD pallet, taken from a study report on the evaluation of space station assembly tasks utilizing the FTS [13], is shown in Table 2, where the steps of the primary worksystem (the FTS) are separated and proceed on different lines from those of the secondary work system (the SSRMS in this case).

Table 2
Task Scripting Example

Secondary Work Systems	Primary Work System
SSRMS unstows PMAD pallet from STS payload bay	--
SSRMS positions pallet in FTS work envelope	--
--	FTS deploys starboard PMAD legs
--	--

The scripts begin with a list of assumptions and end with a list of effects on the work systems used to deploy the FTS. Also listed are any modifications to the original task steps required to perform the task by the FTS.

6.2 Task Modeling

The results of scripting become the basis for further analysis using mathematical modeling and simulation. Parametric studies were conducted to identify requirements, computer graphics models validated such things as access and collision avoidance, and a scale model was used to rehearse the script.

Computer graphics were used to validate three of the assembly tasks that had been scripted:

- o The PMAD installation
- o The Thermal Control Subsystem (TCS) condenser module and radiator installation
- o The inspection of the node to stringer final assembly

The IGRIP computer programming system was used on a Silicon Graphics IRIS 4D/70 GT computer. The simulations provided three-dimensional kinematic models of the FTS Tinman, the RMS, and the Assembly Work Platform Astronaut Translation Devices (ATDs). The simulations also contained geometric models of the Space Station Freedom elements and the STS. A feature of this approach that proved useful was the ability to obtain different views of the operations: for example, a view as seen by the operator using the FTS cameras; a view from the STS Aft Flight Deck; and a global, bird's eye view away from the STS.

An example of the computer graphics is given in Figure 9 where the FTS is shown deploying the PMAD leg. Note that the FTS is attached to the ATD by a Power and Data Grapple Fixture (PDGF). The FTS stabilizes itself by holding onto the PMAD pallet while the RMS holds and maneuvers the pallet for the FTS to deploy the legs.

6.3 Task Evaluation and Ranking

The FTS assembly tasks were ranked on a scale from 0 to 10 in terms of their expected impacts to the following list of attributes:

- o Resources required
- o EVA time saved
- o Reduction in EVA hazard exposure
- o Task complexity
- o Task criticality
- o Task recurrence
- o Task similarity to baseline FTS tasks

Since these factors cannot be measured on a common scale, a ranking methodology capable of dealing with multiple factors was used. The Keeney-Raiffa Multiple Attribute Decision Analysis (MADA) was chosen.

7. Conclusion

The FTS promises to be a useful, reliable, and safe tool to assist the astronauts in performing assembly, maintenance, servicing, and inspection tasks on Space Station

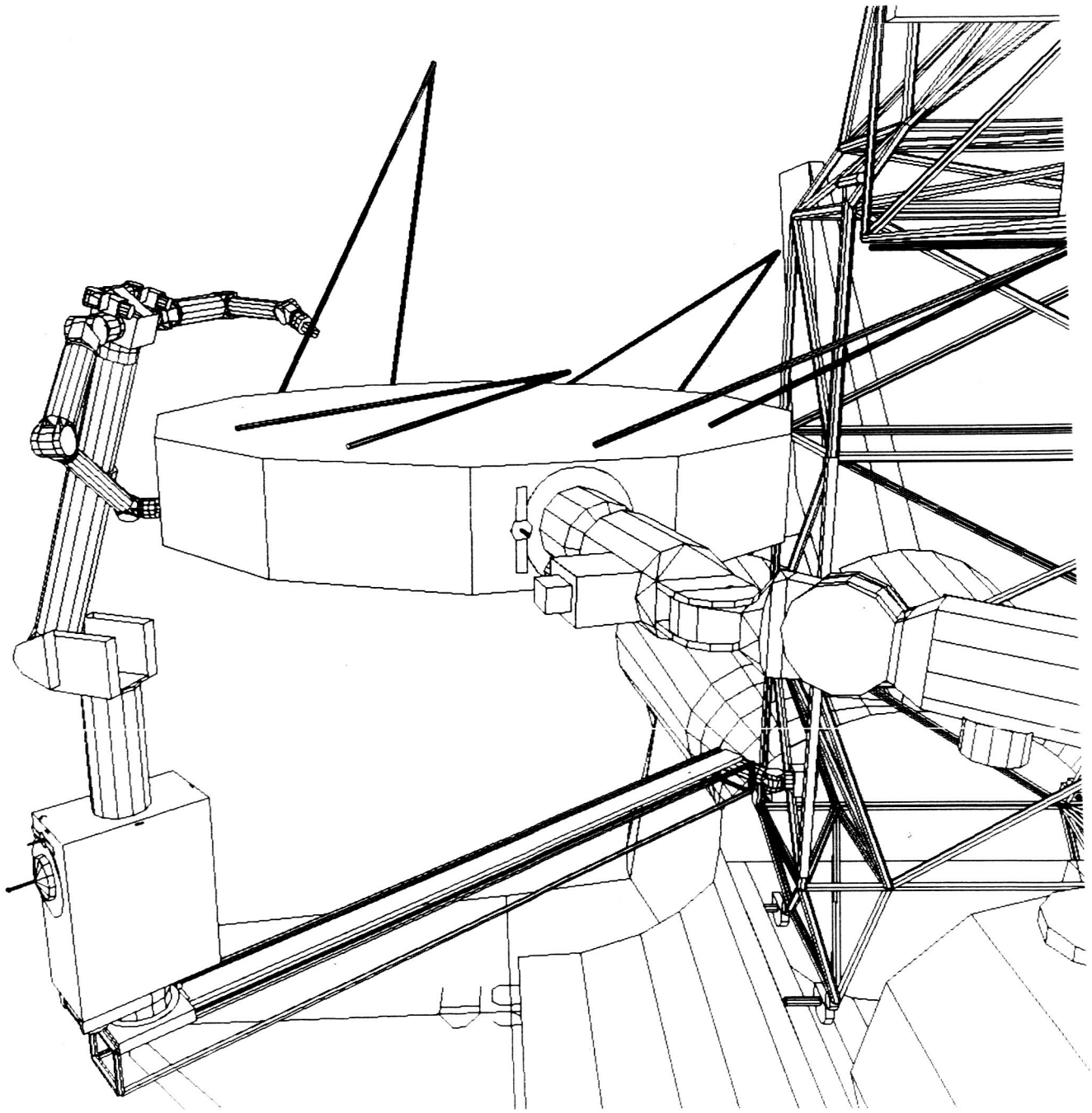


Figure 9. FTS Deploys PMAD Pallet Leg

Freedom and the NSTS. The design challenges have been identified and operational scenarios and task planning have been addressed by the NASA Phase B study team, while candidate designs were being developed by Grumman and Martin Marietta in their Phase B studies.

Progress has been made in identifying the FTS accommodations on Space Station Freedom to successfully integrate the FTS into the program as a useful tool for the space station crew. Commonality and accessibility are primary considerations in the selection of utility ports and structural attachment points for the FTS.

A structured task analysis methodology has been developed, and it is being used to construct scenarios of FTS assembly tasks. With this technique, the FTS Tinman was shown to be capable of executing a wide range of tasks, contingent upon the provision of certain resources and associated work system capabilities.

FTS is unique in that it will be required to operate in a much less structured environment than previously developed industrial robots. It will be required to perform many varied tasks with varying precision throughout its expected lifetime. Because these tasks will increase in complexity, the system must be capable of substantial growth and evolution. It is a program that focuses more on the future than the present technology.

8. Acknowledgments

The information presented in this paper has been obtained from numerous sources. The major contributors were:

- Engineers from--Goddard Space Flight Center, Johnson Space Center, the Jet Propulsion Laboratory, Marshall Space Flight Center, Ames Research Center, Langley Research Center, the National Institute of Standards and Technology, and the Oak Ridge National Laboratories.

- Support contractors--Advanced Technology & Research (ATR), Operations Research, Inc. (ORI), Engineering & Economics Research, Inc. (EER), ST Systems Corp. (STX), OAO Corp., Ocean Systems Engineering (OSE), and CTA.

- Universities--Dr. Richard Volz at the University of Michigan; Dr. Pradeep Khosla at Carnegie Mellon University; and Dr. Ed Haug at the University of Iowa. Dr. Dave Criswell at the University of California at San Diego headed a team of researchers from the Consortium for Space and Terrestrial Automation and Robotics (CSTAR) who included: Dr. George Kondraske at the University of Texas at Arlington; Dr. Michael Walker at the University of Michigan; Dr. Ken Lauderbaugh at Rensselaer Polytechnic Institute; and Dr. Kai-Hsiung Chang at Auburn University.

- Computer graphics--Tim Carnahan, GSFC; Eugene Aronne, ATR; and Andy Cooper, ORI.

9. References

[1] Flight Telerobotic Servicer Requirements Document for Definition and Preliminary Design, GSFC Space Station (SS) document, SS-GSFC-0028, April 1987.

[2] Hinkal S., J. Andary, J. Watzin, and D. Provost, "The Flight Telerobotic Servicer (FTS): A Focus for Automation and Robotics on the Space Station," Acta Astronautica, Vol. 17, No. 8, pp. 759-768, 1988.

[3] Flight Telerobotic Servicer Strawman Concept Engineering Report, SS-GSFC-0031, March 15, 1987.

[4] Flight Telerobotic Servicer Requirements Document for Phase C/D, SS-GSFC-0043, October 27, 1988.

[5] Flight Telerobotic Servicer Tinman Concept In-house Phase B Study Final Report, Volumes I and II, SS-GSFC-0042, September 9, 1988.

[6] Albus J., H. McCain, and R. Lumia, "NASA/National Bureau of Standards (NBS) Standard Reference Model for Telerobot Control System Architecture (NASREM)," SS-GSFC-0027, December 4, 1986.

[7] Space Station Design Criteria and Practices for Accommodation of Robotic Systems, Space Station Freedom Program (SSP) document, SSP TBD.

[8] Space Station Man-Systems Integration Standards, NASA-STD-3000, Vol. IV

[9] Space Station Program Definition and Requirements Document, SSP 30000.

[10] Change Request BG 030082A to SSP 30000, Sec. 3, Rev. G, (Tables 3-26 and 3-27), approved December 14, 1988.

[11] Flight Telerobotic Servicer Task Analysis, Volumes I and II, draft revision H, December 15, 1988.

[12] Space Station Assembly Operations, Functional Flows and Resource Allocations, Flights MB-2 through MB-5, Space Station Freedom Program Office presentation, June 1988.

[13] Study Report, Evaluation of Space Station Freedom Assembly Tasks Utilizing the Flight Telerobotic Servicer, FTS Mission Utilization Team, December 15, 1988.

THE FLIGHT TELEROBOTIC SERVICER: FROM FUNCTIONAL ARCHITECTURE TO COMPUTER ARCHITECTURE

RONALD LUMIA
JOHN FIALA
ROBOT SYSTEMS DIVISION
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

ABSTRACT

The NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) has been adopted by NASA for use in the Flight Telerobotic Servicer (FTS), a two armed telerobotic manipulator which will build and maintain the Space Station. NASREM is technology independent; the same functions must be performed by all controllers. NASREM provides the paradigm which allows the FTS to evolve with technology because standard interfaces can be defined so that functionally equivalent software and hardware modules may be interchanged. After a brief tutorial on the NASREM functional architecture, the approach to its implementation will be shown. First, interfaces must be defined which are capable of supporting the known algorithms. This will be illustrated by considering the interfaces required for the SERVO level of the NASREM functional architecture. After interface definition, the specific computer architecture for the implementation must be determined. This choice is obviously technology dependent. An example illustrating one possible mapping of the NASREM functional architecture to a particular set of computers which implements it will be shown. The result of choosing the NASREM functional architecture is that it provides a technology independent paradigm which can be mapped into a technology dependent implementation capable of evolving with technology in the laboratory as well as in space.

INTRODUCTION

The requirements of the Flight Telerobotic Servicer (FTS) of the Space Station are driving the development of robot systems for space applications. One of the key requirements is that the telerobot should be able to evolve with technology. This requirement implies the need for a reference model or functional architecture for the control system. A functional architecture is essential for several reasons. The control system cannot be developed as a static system but must be conceived to be able to evolve over time in order to benefit from advances in technology. Consequently, the architecture must be sufficiently flexible to support telerobotics in the beginning of the program and to

gradually support more autonomy of robot tasks. NASREM provides this functional architecture. Another aspect compelling the use of NASREM is that it provides a common reference model to which all designs must interface. Previous work in the Automated Manufacturing Research Facility (AMRF) at the National Institute of Standards and Technology (formerly NBS) has shown that system integration is the most difficult challenge [1]. The value associated with such a standard means that there is a common basis for the comparison of different design approaches for solving technical problems.

While the choice of a functional architecture is a crucial decision for the evolvability of the FTS, it does not contain all of the information required for a complete design. The purpose of this paper is to illustrate how a designer would proceed from the functional architecture to the functioning FTS. This paper is organized in the following manner to delineate the design process from conception to realization. First, a description of the NASREM reference model is presented. Then, the method used to define the interfaces is presented. This is followed by an example of how a particular computer architecture can realize the functional architecture. Finally, the potential impact of NASREM on both space and terrestrial applications of robots is assessed.

NASA/NBS STANDARD REFERENCE MODEL FOR TELEROBOT CONTROL SYSTEM ARCHITECTURE (NASREM)

The fundamental paradigm of the control system is shown in Figure 1. The control system architecture is a three legged hierarchy of computing modules, serviced by a communications system and a global memory. The task decomposition modules perform real-time planning and task monitoring functions; they decompose task goals both spatially and temporally. The sensory processing modules filter, correlate, detect, and integrate sensory information over both space and time in order to recognize and measure patterns, features, objects, events, and relationships in the external world. The world modeling modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in global memory. Global memory is a database which contains the system's best estimate of the state of the external world. The world modeling modules keep the global memory database current and consistent.

The first leg of the hierarchy consists of task decomposition modules which plan and execute the decomposition of high level goals into low level actions. Task decomposition involves both a temporal decomposition (into sequential actions along the time line) and a spatial decomposition (into concurrent actions by different subsystems). Each task decomposition module at each level of the hierarchy consists of a job assignment manager, a set of planners, and a set of executors.

The second leg of the hierarchy consists of world modeling modules which model and evaluate the state of the world. The "world model" is the system's best estimate and evaluation of the history, current state, and possible future states of the world, including the states of the system being controlled. The "world model" includes both the world modeling modules and a knowledge base stored in a global memory database where state variables, maps, lists of objects and events, and attributes of objects and events are maintained. The world model maintains the global memory knowledge base by accepting information from the sensory system, provides predictions of expected sensory input to the corresponding sensory system modules, based on the state of the task and estimates of the external world, answers "What is?" questions asked by the executors in the corresponding task decomposition modules, and answers "What if?" questions asked by the planners in the corresponding task decomposition modules.

The third leg of the hierarchy consists of sensory system modules. These recognize patterns, detect events, and filter and integrate sensory information over space and time. The sensory system modules at each level compare world model predictions with sensory observations and compute correlation and difference functions. These are integrated over time and space so as to fuse sensory information from multiple sources over extended time intervals. Newly detected or recognized events, objects, and relationships are entered by the world modeling modules into the world model global memory database, and objects or relationships perceived to no longer exist are removed. The sensory system modules also contain functions which can compute confidence factors and probabilities of recognized events, and statistical estimates of stochastic state variable values.

The control architecture has an operator interface at each level in the hierarchy. The operator interface provides a means by which human operators, either in the space station or on the ground, can observe and supervise the telerobot. Each level of the task decomposition hierarchy provides an interface where the human operator can assume control. The task commands into any level can be derived either from the higher level task decomposition module, from the operator interface, or from some combination of the two. Using a variety of input devices, a human operator can enter the control hierarchy at any level, at any time of his choosing, to monitor a process, to insert information, to interrupt automatic operation and take control of the task being performed, or to apply human intelligence to sensory processing or world modeling functions.

The sharing of command input between human and autonomous control need not be all or none. It is possible in many cases for the human and the automatic controllers to simultaneously share control of a telerobot system. For example, in an assembly operation, a human might control the position of an end effector while the robot automatically controls its orientation.

INTERFACE DEFINITION

In order to implement a functional architecture, especially one like NASREM which allows evolution with technology, the interfaces must be carefully defined. Although the NASREM functional architecture specifies the purpose of each module in the control system hierarchy, it does not completely specify the interfaces between modules. This section will describe the method by which the interfaces for the SERVO level of the hierarchy have been defined. The method involves gathering all of the algorithms available for SERVO level control, dividing each algorithm into the parts which inherently belong to task decomposition, world modeling, and sensory processing, and then deriving the interfaces which will support these algorithms.

The NASA/NBS Standard Reference Model (NASREM) Telerobot Control System Architecture, as presented in [2], defines the basic architecture for a robot control system capable of teleoperation and autonomy in one system. Recently, efforts have been directed at specifying in detail the architecture requirements for robotic manipulation. An important criterion for the design is that it support the algorithms for manipulator control found in the literature. This assures that the control system can serve as a vehicle for evaluating algorithms and comparing approaches. Any design, however, must constrain the problem sufficiently so that detailed interfaces can be devised.

With this in mind, the Servo Level design was based on a fundamental control approach which computes a motor command as a function of feedback system state y, desired state (attractor) y_d, and control gains. In this approach, the gains are coefficients of a linear combination of state errors $(y-y_d)$. The system state and its attractor are composed from the physical quantities to be controlled, (i.e. position, force, etc.,) and can be expressed in an arbitrary coordinate system. This type of algorithm is the basis for almost all manipulator control schemes [3]. However, this basic algorithm is inadequate for controlling the gross aspects of manipulator motion, as described in [8]. The servo algorithm can provide "small" motions so that the algorithm's transient dynamics are not significant in shaping the gross motion. This means that the Primitive Level must generate the gross motion through a sequence of inputs to the Servo Level. This can be achieved through an appropriate sequence of either attractor points [3,4] or gain values [8].

Figure 2 depicts the detailed Servo Level design. The task decomposition module at the Servo Level receives input from Primitive in the form of the command specification parameters. The command parameters include a coordinate system specification C_z which indicates the coordinate system in which the current command is to be executed. C_z can specify joint, end-effector, or

Cartesian (world) coordinates. Given with respect to this coordinate system are desired position, velocity, and acceleration vectors (z_d, z_d, z_d) for the manipulator, and the desired force and rate of change of force vectors (f_d, f_d). These command vectors form the attractor set for the manipulator. The K's are the gain coefficient matrices for error terms in the control equations. The selection matrices (S,S') apply to certain hybrid force/position control algorithms. Finally, the "Algorithm" specifier selects the control algorithm to be executed by the Servo Level.

When the Servo Level planner receives a new command specification, the planner transmits certain information to world modeling. This information includes an attention function which tells world modeling where to concentrate its efforts, i.e. what information to compute for the executor. The executor simply executes the algorithm indicated in the command specification, using data supplied by world modeling as needed.

The world modeling module at the Servo Level computes model-based quantities for the executor, such as Jacobians, inertia matrices, gravity compensations, Coriolis and centrifugal force compensations, and potential field (obstacle) compensations. In addition, world modeling provides its best guess of the state of the manipulator in terms of positions, velocities, end-effector forces and joint torques. To do this, the module may have to resolve conflicts between sensor data, such as between joint position and Cartesian position sensors.

Sensory processing, as shown in Figure 2, reads sensors relevant to Servo and provides the filtered sensor readings to world modeling. In addition, certain information is transmitted up to the Primitive Level of the sensory processing hierarchy. Primitive uses this information, as well as information from Servo Level world modeling, to monitor execution of its trajectory. Based on this data, Primitive computes the stiffness (gains) of the control, or switches control algorithms altogether. For example, when Primitive detects a contact with a surface, it may switch Servo to a control algorithm that accommodates contact forces.

A more complete description of the Servo Level is available in [3] where the vast majority of the existing algorithms in the literature are described. The same process for developing the interfaces based on the literature has also been performed for the Primitive level and is available in [4]. While the procedure is planned for each level in the hierarchy, the amount of literature support tends to decrease as one moves up the hierarchy.

EXAMPLE OF A COMPUTER ARCHITECTURE TO IMPLEMENT NASREM

Once the interfaces are defined, it is possible to choose a

computer architecture and begin to realize the system. This section will describe the specific implementation under construction at NIST. While every effort is being made to do the job properly, there is no reason to assume that this implementation is optimal in any way. It is simply illustrates one realistic method to implement the NASREM architecture.

While a functional architecture is technology independent, its implementation obviously depends entirely on the state-of-the-art of technology. The designer must choose existing computers, buses, languages, etc., and, from these tools, produce a computer architecture capable of performing the functions of the functional architecture. The system must adequately meet the real-time aspects of the controller so that adequate performance is achieved through careful consideration of computer choice, multiple processor real-time operating system, inter-processing communication requirements, tasking within certain processors, etc. For a more detailed description of this methodology, see [5].

The NIST implementation considers two aspects of the process: the development environment on which the code is developed, debugged, and tested as well as possible, and the target environment where the code for the real-time robot control system is executed. Figure 3 shows the approach. A network of SUN workstations running UNIX is used for the development environment, sacrificing the speed of the developed code for the ease of development. Once the code is tested as well as possible, it is downloaded to the target system. The target system consists of a VME backplane of several (currently 6) Motorola 68020 processors. For rapid iconic image processing, the PIPE system [6] is interfaced. The target hardware drives the Robotics Research Corporation arm.

From the software side, the multiprocessing operating system used for the target is required to be as simple as possible so that the overhead is minimized. The duties of the operating system are limited to very simple actions such as downloading and starting up the processors and interprocessor communication. Tasking is not performed at the lower levels of the hierarchy because of the overhead associated with context switches. NIST researchers are currently investigating three alternatives for tasking: tasking provided by the native compiler, pSOS tasking, and ADA tasking. Interprocessor communications alternatives including pRISM, sockets, etc., must also be evaluated empirically. The actual application code is written in ADA. Although ADA compilers usually cannot currently produce code as efficient as other languages such as C, NIST researchers have shown that the gap is steadily decreasing [7].

The application code is developed by programming the processes which achieve the functions associated with the boxes in the functional architecture. The problem then becomes one of assigning each of the processes, such as those shown in Figure 2,

to a particular processor. There is a clear trade-off between the cost of the solution and the performance of the system. There are currently no software tools which automatically perform this assignment based on an arbitrary index of performance. The approach at NIST is step-wise refinement of the performance of the system. Given the particular hardware being used, a certain number of processors is chosen arbitrarily. For that configuration, the processes are assigned to the processors. Then, the system is evaluated in terms of its performance. If the performance is unacceptable, the designer has several options. The first option is to add more processors. This alternative is balanced against the possibility of the additional communication requirements of the processors. Another alternative is to add faster processors or special purpose processors, such as dynamics chips, which optimize particularly compute intensive operations. This trade-off clearly relates to cost. Another alternative is to reassign the processes to the processors in order to balance the workload of each processor. Each of the alternatives can be used by the designer in order to improve the performance of the system. This allows a particular configuration which implements the functional architecture to change with time as improvements in technology are realized.

CONCLUSION

The NASREM functional architecture provides the technology independent paradigm which serves as the foundation from which any NASREM implementation can be derived. Interfaces may be developed for the NASREM architecture which will take into account the research already published in the literature. When a NASREM implementation is desired, the result is, by necessity, a reflection of the current state-of-the-art. However, since the interfaces are carefully specified, alternative software and hardware solutions may easily be tested and integrated. This will allow the FTS to evolve with technology, both for space as well as for terrestrial applications.

REFERENCES

[1] J.A. Simpson, R.J. Hocken, J.S. Albus. "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing Systems, 1, 1, 1982, 17.

[2] J.S. Albus, R. Lumia, H.G. McCain, "NASA/NBS Standard Reference Model For Telerobot Control System Architecture (NASREM)," NBS Technote #1235, also as NASA document SS-GSFC-0027.

[3] J. Fiala, "Manipulator Servo Level Task Decomposition," NIST Technote #1255, April 20, 1988.

[4] A.J. Wavering, "Manipulator Primitive Level Task Decomposition," NIST Technote #1256, January 5, 1988.

[5] J. Michaloski,T. Wheatley,and R. Lumia, "Timing Analysis for a Parallel Pipelined Hierarchical Control System", 9th Real-Time Systems Symposium, (submitted).
[6] E.W. Kent, M.O. Shneier, and R. Lumia, "PIPE," Journal of Parallel and Distributed Computing, Vol. 2, 1985, pp. 50-78.
[7] S. Leake, "A Comparison of Robot Kinematics in ADA and C on Sun and microVAX," Robotics and Automation Session, IASTED, Santa Barbara, CA., May 25-27-,1988.
[8] J. Fiala, "Generation of Smooth Trajectories without Planning," 1989 IEEE Robotics and Automation Conference, (submitted).

NASREM: NASA/NBS STANDARD REFERENCE MODEL

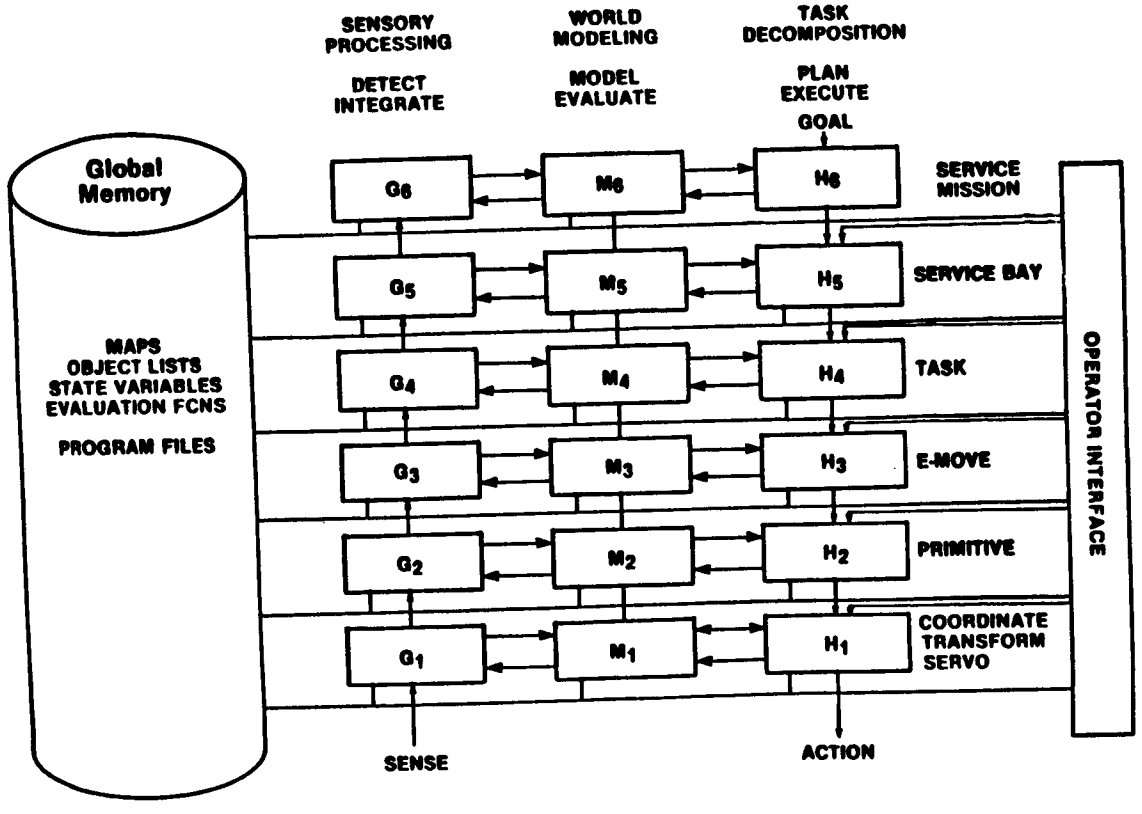


FIGURE 1

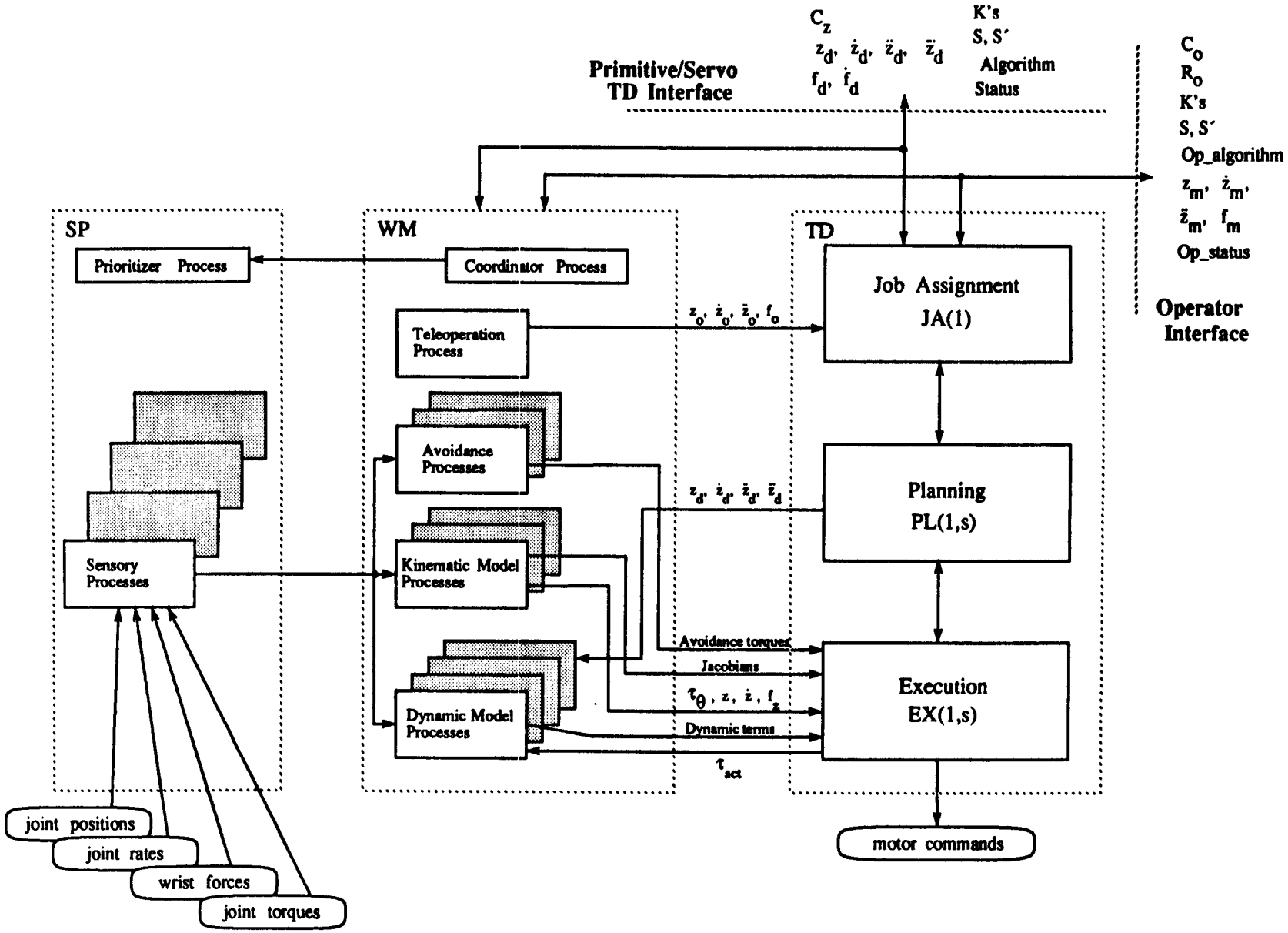

Primitive/Servo
TD Interface
C_z
z_d, $\dot{z}_d$, $\ddot{z}_d$, $\bar{z}_d$
f_d, $\dot{f}_d$
K's
S, S'
Algorithm
Status
C_o
R_o
K's
S, S'
Op_algorithm
z_m, $\dot{z}_m$,
$\ddot{z}_m$, f_m
Op_status
Operator
Interface
SP
WM
TD
Prioritizer Process
Coordinator Process
Job Assignment
JA(1)
Teleoperation
Process
z_o, $\dot{z}_o$, $\ddot{z}_o$, f_o
Avoidance
Processes
Planning
PL(1,s)
z_d, $\dot{z}_d$, $\ddot{z}_d$, $\bar{z}_d$
Sensory
Processes
Kinematic Model
Processes
Avoidance torques
Jacobians
τ_θ, z, $\dot{z}$, f_z
Execution
EX(1,s)
Dynamic Model
Processes
Dynamic terms
τ_{act}
joint positions
joint rates
wrist forces
joint torques
motor commands

FIGURE 2

SYSTEM DEVELOPMENT
(View at Hardware)

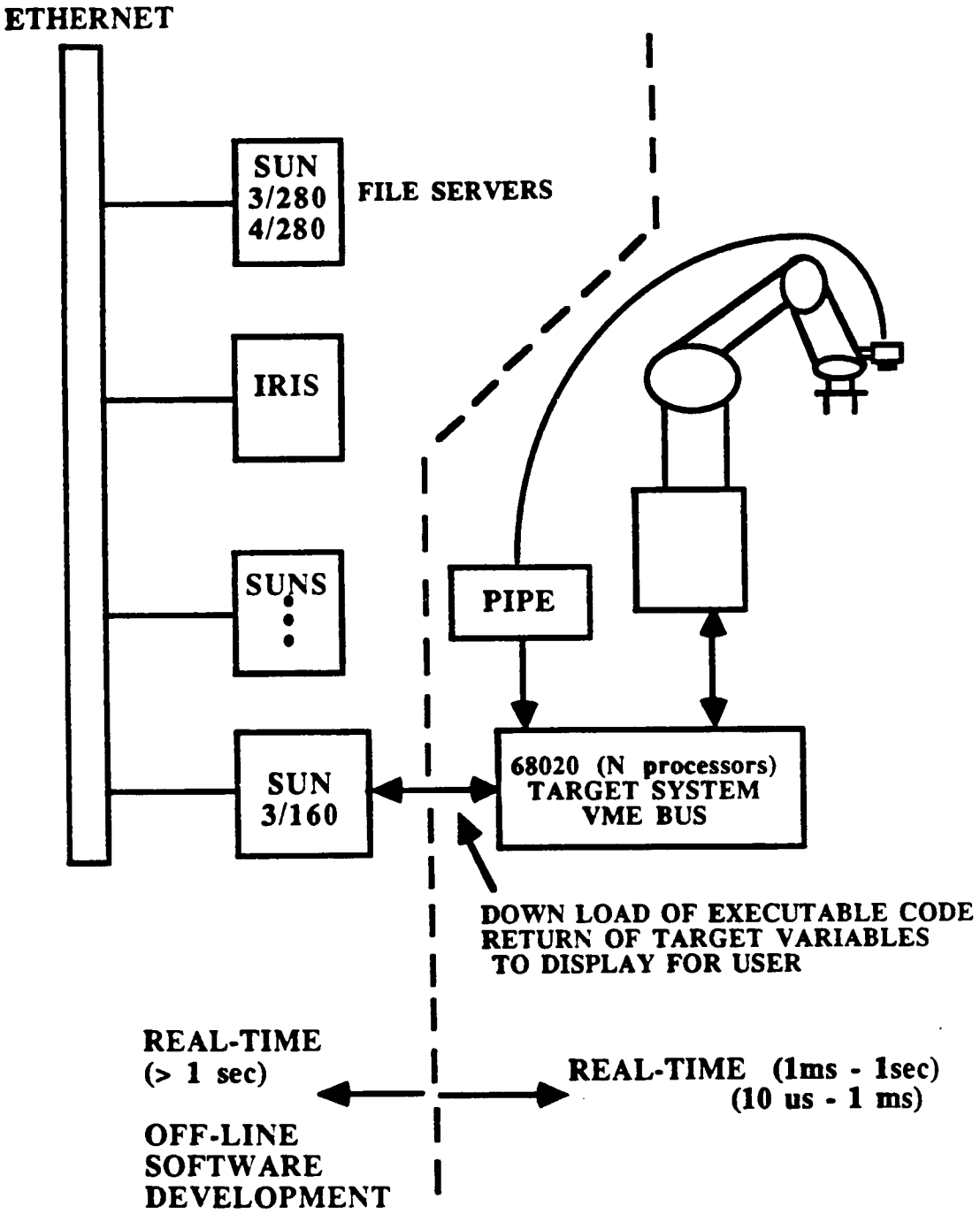


FIGURE 3

RESEARCH AND DEVELOPMENT ACTIVITIES AT THE GODDARD SPACE FLIGHT CENTER FOR THE FLIGHT TELEROBOTIC SERVICER PROJECT

Stanford Ollendorf, Chief

Office of Telerobotic Engineering
Goddard Space Flight Center
Greenbelt, Maryland 20771

1. Introduction

The Flight Telerobotic Servicer (FTS) is being developed by the Goddard Space Flight Center (GSFC) for performing a variety of assembly, servicing, inspection and maintenance tasks on the Space Station (Figure 1). The Project Office at GSFC has tasked the Engineering Directorate to assemble a robotics research and development program which will support the FTS project. The activities center around support for the Development Test Flight (DTF) on the Space Shuttle and investigations of operational problems associated with the FTS on Space Station Freedom. For the DTF, areas such as control algorithms, safety systems, and end-effectors will be developed. For FTS operations, the emphasis will be to develop a dual-arm bi-lateral force-reflecting teleoperator and use it as an FTS Operational Simulator (FTSOS). The simulator will be used to investigate operational techniques, camera configurations, operator interfacing, orbital replacement unit (ORU) designs, end-effector designs, and training techniques. After a series of test activities, reports will be generated for input to the DTF and FTS designs.

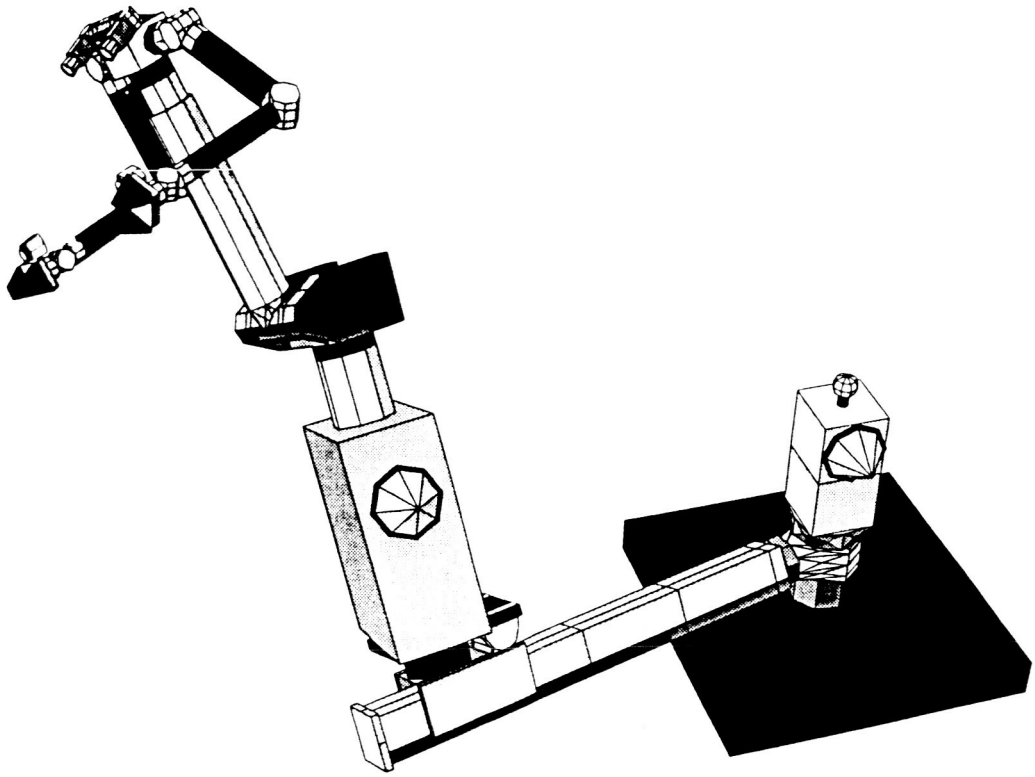


Figure 1. FTS design concept.

2. Facility Description

In support of this effort a robotic facility is being established at GSFC. This facility will be used to develop, test, integrate and evaluate new robotic technologies required to support the FTS Project (Figure 2). It will contain a gantry robot with six degrees of freedom capable of lifting up to 4000 pounds of payload and applying 4000 ft. pounds of torque as well. Suspended from one mast of the gantry will be a set of six degree of freedom industrial arms, which will be used as an FTS Operational Simulator (FTSOS). The other mast will carry a grapple to emulate the Space Station Remote Manipulator System (SSRMS) and will be used primarily to transport payloads to and from the worksite. An operator workstation, installed in a mockup of the STS Aft Flight Deck (AFD) mockup, will permit teleoperation in the constrained environment of the Space Shuttle. This AFD will be designed to be reconfigurable in order to determine the best positioning of hand controllers and displays.

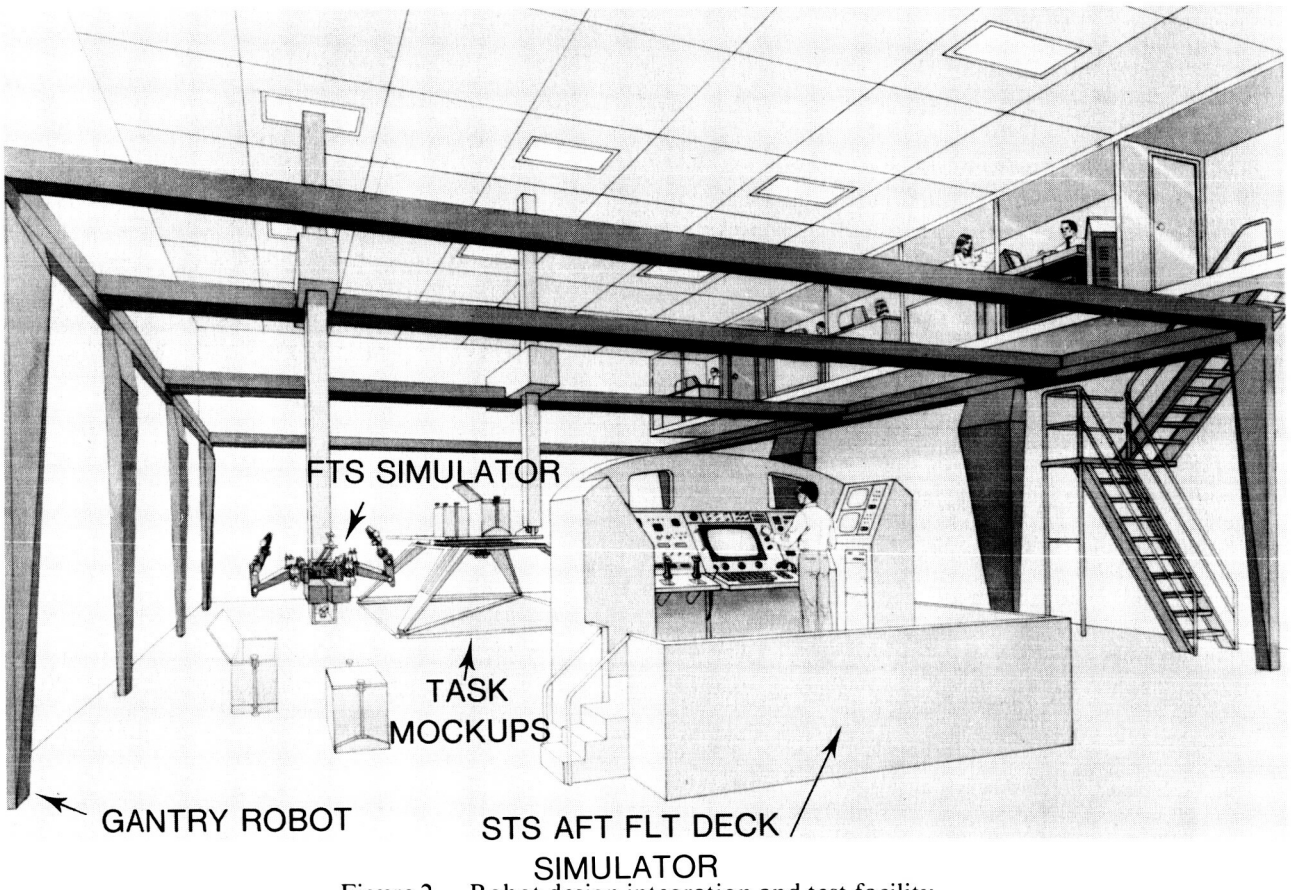


Figure 2. Robot design integration and test facility.

As an adjunct to the facility, a technology is being developed for graphically displaying each move of the robot in performing its tasks. The graphics are being used to determine such things as the robot's reach capability, and check for collision avoidance. The graphic simulator enables the tasks to be defined by breaking down each task into sub-steps from which a "script" can be created. The script allows for the creation of a model representation of the FTS and its relationship to the assembly phase of Space Station. Currently the capability incorporates the inverse kinematics associated with the robot motion. Eventually through research being performed at the University of Iowa [1], dynamic models will be developed and integrated into the system for improved representation.

Working closely with the National Institute of Standards and Technology (NIST), a computer architecture is being established which allows for incremental development and evolution of the telerobotic system leading to

greater autonomy. The NASA Standard Reference Model or NASREM [2], has been selected for implementation into the facility. Adopting this architecture, which NIST hopes to standardize, will link the NASA developments to U.S. Industry, making technology transfer possible.

In addition to the gantry and FTS Operational Simulator, a seven degree of freedom industrial manipulator system (Figure 3), will be combined with a six degree mini-master controller to investigate safety and control problems associated with the operation of this complex system [3]. As data is derived from this test bed, it will be made available to the FTS contractor for their use in designing a flight system. As an adjunct to the operational test facility, smaller industrial or research robots are being used for pre-cursor checkout of end-effectors, software development and technologies arriving from other NASA research facilities (Figure 4).

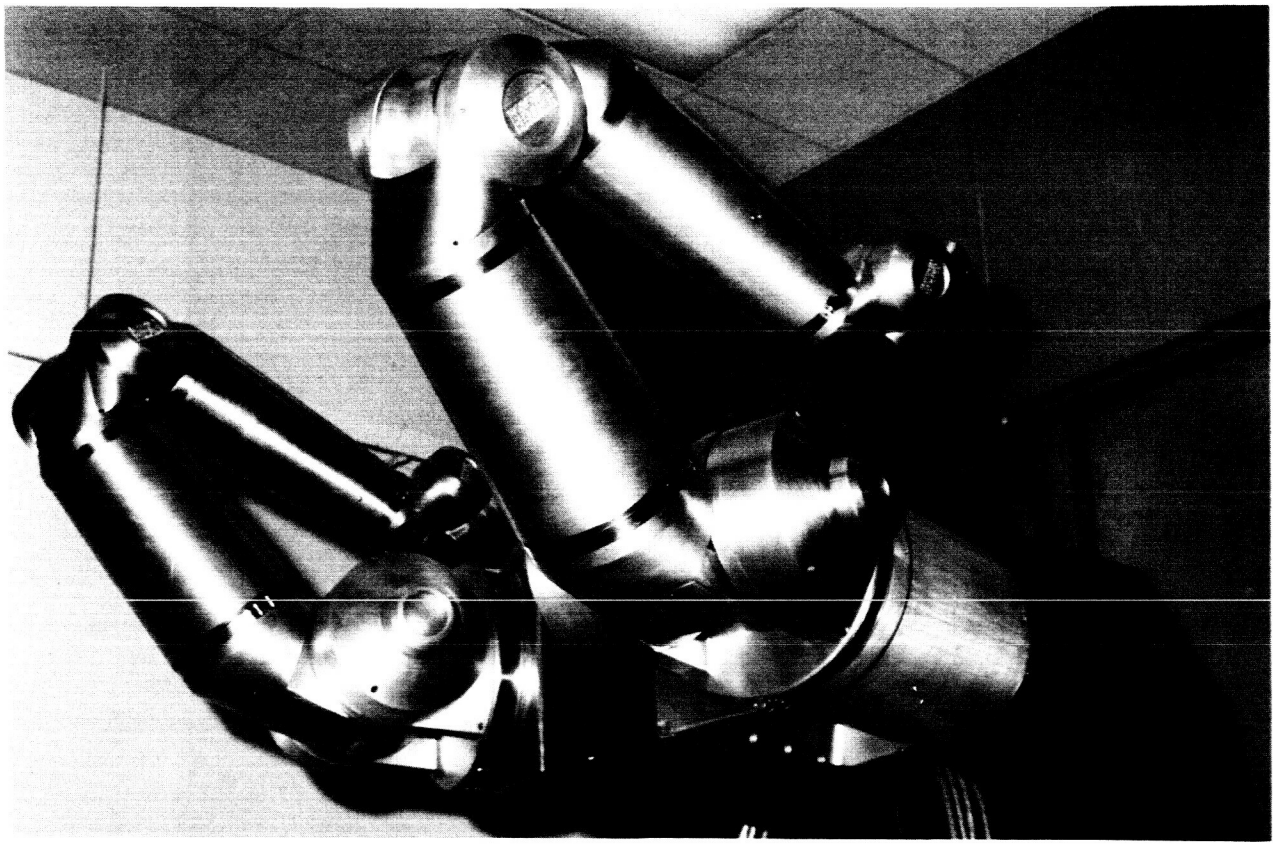


Figure 3. Seven degree of freedom industrial manipulator.

In order to reduce the overall complexity of the robot, that is having to include sophisticated vision recognition systems, dexterous hands, etc. ... the robotic task must be "friendly" in its design. As part of this activity, GSFC is developing structures and mechanisms which will interface with the robot in a known pre-determined manner. Examples of these features are handles which mate with ordinary parallel jaw grippers, singularly actuated orbital replacement units (ORU) and common utility connectors. Figures 5 and 6 show "robot friendly" structural attachments and an ORU with low torque and force "J" hook actuators. Although it is recognized that all tasks cannot be predetermined for the FTS and eventually it will have to operate in a less structured environment, these techniques, when standardized, will make the robot a more cost effective system.

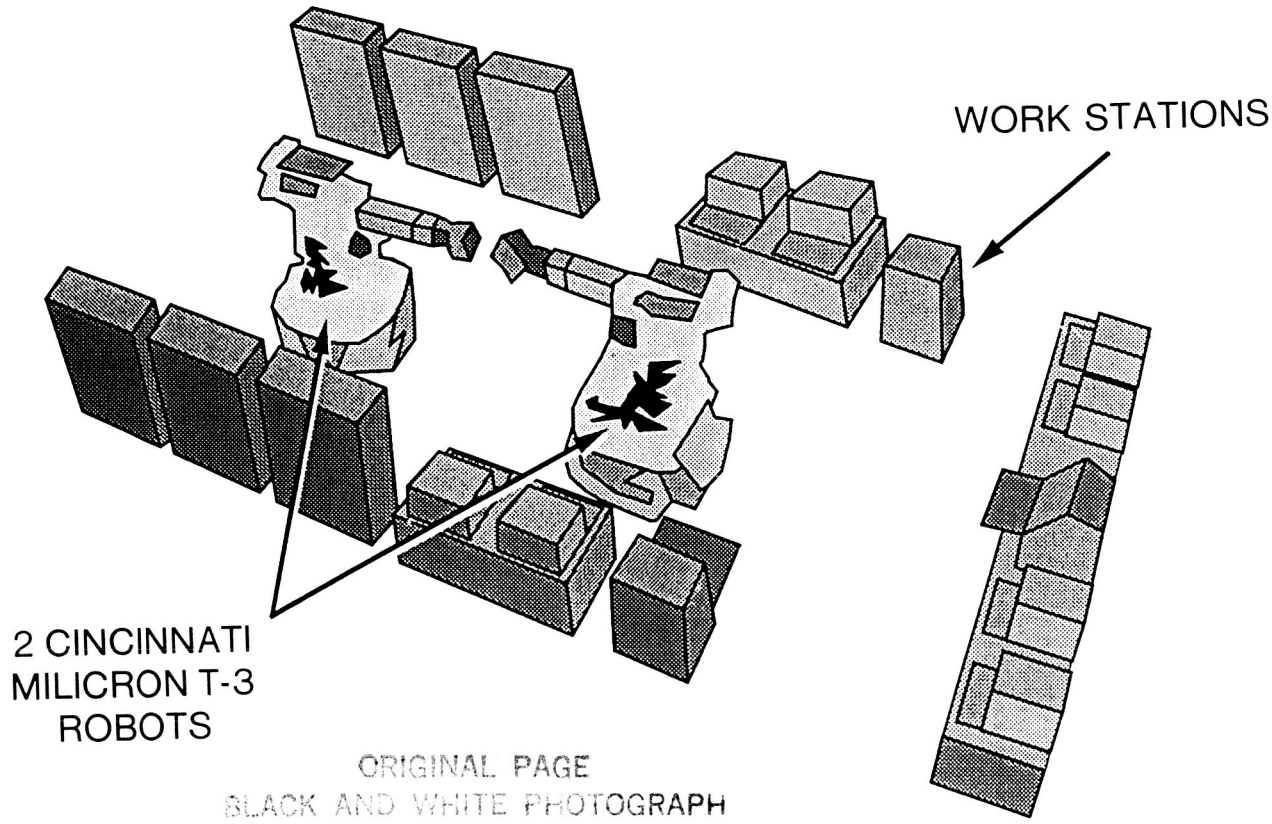


Figure 4. Pre-cursor checkout facility.

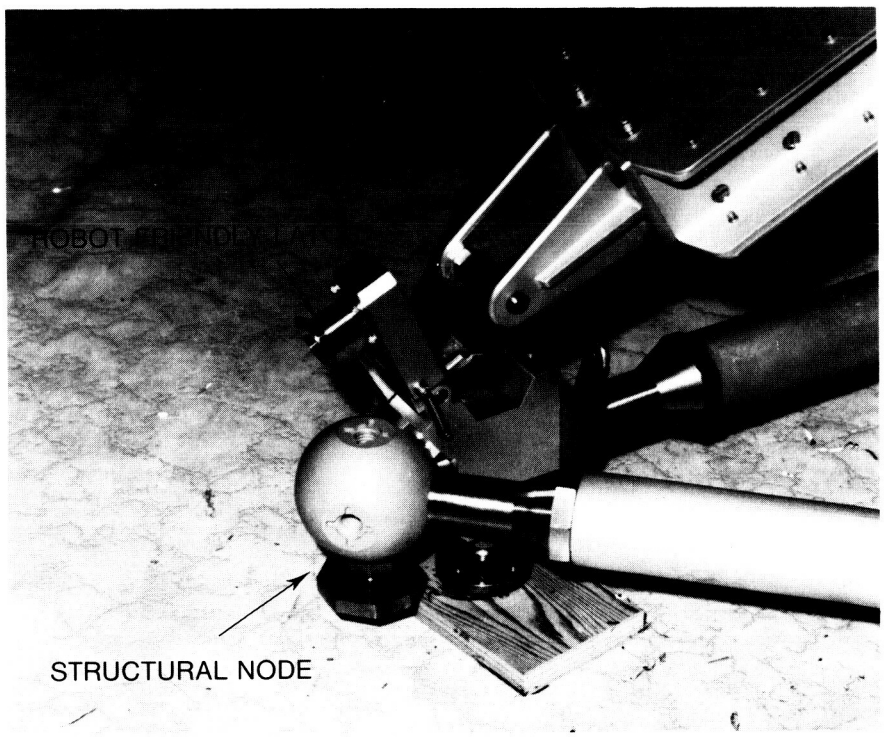


Figure 5. Robot friendly structural attachment.

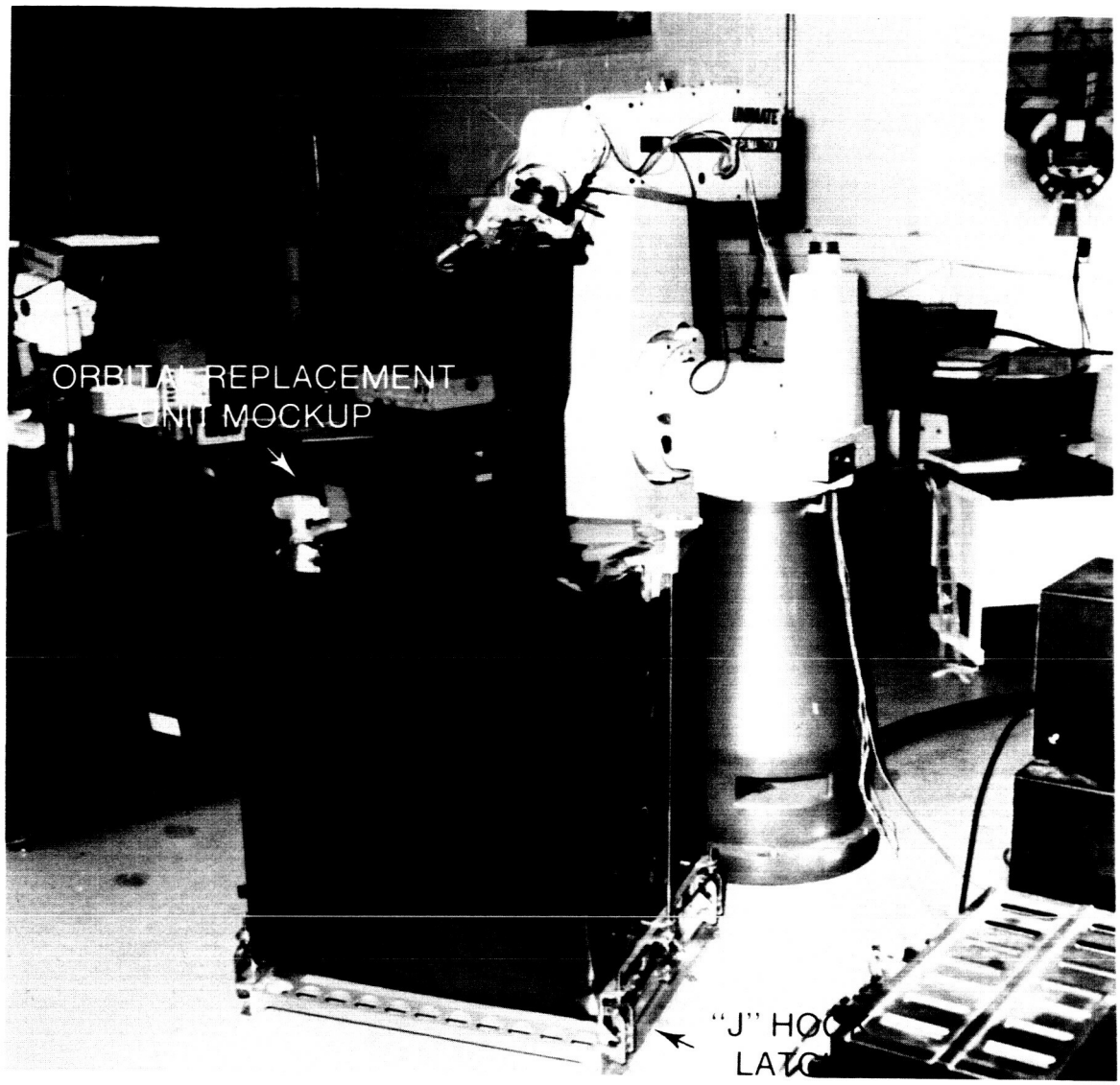


Figure 6. ORU mockup with "J" hook attach mechanisms.

3. Test Program

Each element of the FTS research and development program has been broken down to support specific events in the DTF mission and FTS development (Figure 7). These usually coincide with Preliminary Design Reviews (PDR) or Critical Design Reviews (CDR). In this manner, data accrued from the test program will be available to the FTS design activities in a timely manner. For the first phase, the gantry robot together with two floor mounted PUMA 762 robots, operating through the use of enhanced workstation will:

1. Deploy a Station Interface Adapter (SIA) leg.

2. Attach the SIA to a truss node.

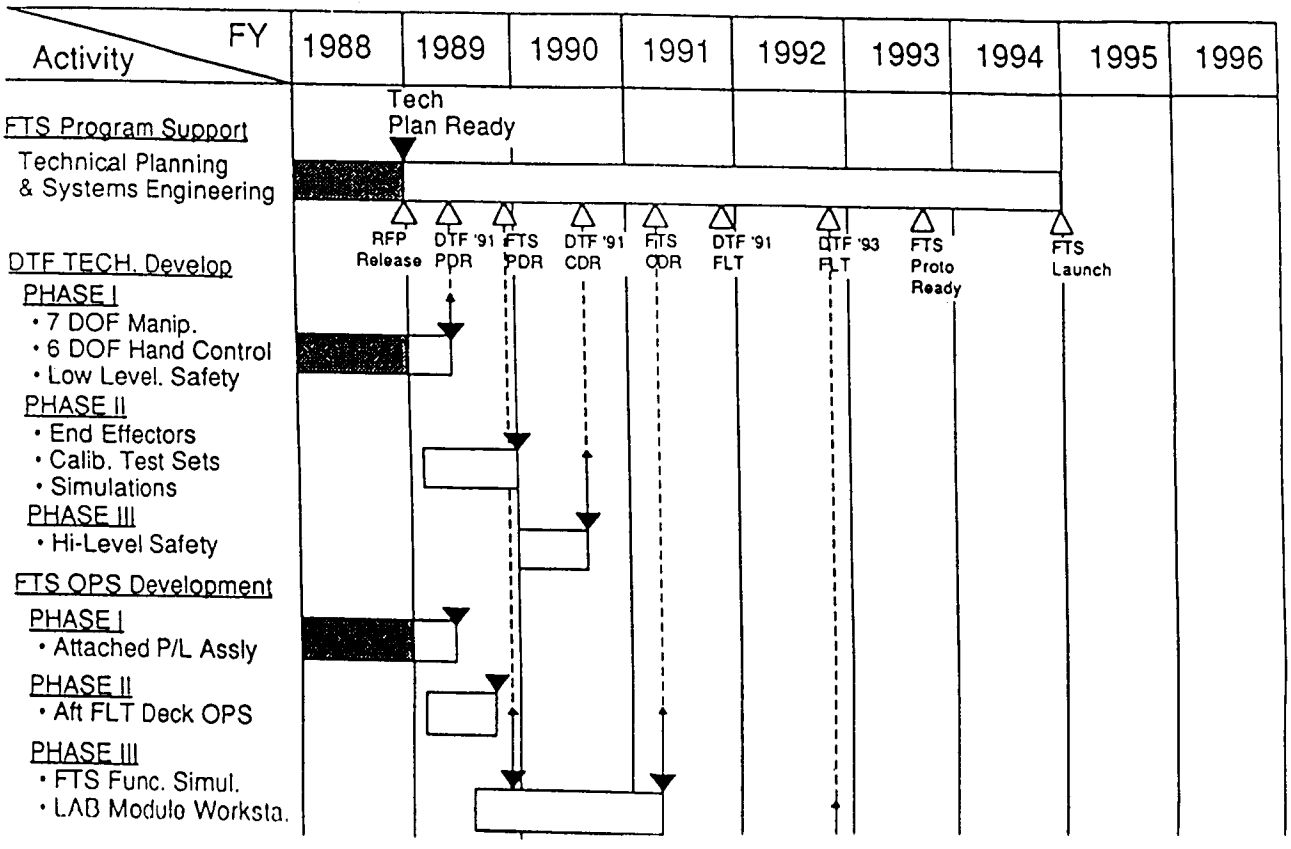


Figure 7. Test program schedule.

3. Perform a Payload Interface Adapter (PIA) actuator closure.

4. Connect a thermal utility connection.

5. Changeout a sub-ORU.

6. Perform a simulated instrument alignment.

For the second phase, a new telerobotic workstation, designed to physically represent the Space Shuttle Aft Flight Deck (AFD) constraints, will be built. The workstation will represent the results of a human engineering study concerning the location of displays and controls. This workstation will be used for future demonstrations. The AFD workstation software will be implemented in ADA.

During the third phase, the dual arm teleoperated manipulator system will become the FTS operational simulator (FTSOS). The FTS operational simulator will have kinematically identical 6-DOF masters and slaves. The teleoperator system will be integrated onto the gantry. A new task mockup representative of the Space Station Electrical Power System (EPS) radiator panel assembly will be built. The activity will consist of inserting the radiator panels into a mockup heat exchanger using the FTSOS system and AFD workstation.

For the DTF technology, seven degree of freedom dual-arm telerobotic controls and a safety testbed will be developed. The testbed will investigate the ADA language, mini-masters, control techniques, end-effector designs, system safety, and dynamic simulations.

During the initial phase, the equipment necessary to build the seven degree of freedom dual-arm manipulator testbed system will be implemented for force-feedback teleoperation. Safety algorithms will be

developed and integrated into the testbed using expert systems where applicable. The dual-arm system, with its control algorithms and safety system imbedded into the NASREM architecture, will be used to investigate teleoperator issues.

For the second phase, end-effectors will be developed and integrated. A calibration task set based on the expected DTF mission will be used for determining end-to-end performance of the hand controller and manipulator system.

4. Future Activities

As the characteristics of the FTS become better understood and its capabilities to perform useful tasks have been demonstrated, the test program will be broadened to include assembly of large Space Station attached payloads, servicing of scientific instruments on earth observation platforms, and the investigation of rendezvous and docking techniques. These activities will be consolidated in an extension of the current facility (Figure 8) planned to be completed in FY 93. Also housed in this extension will be a full sized mockup of a Space Station node and cupola FTS workstation for operational simulation.

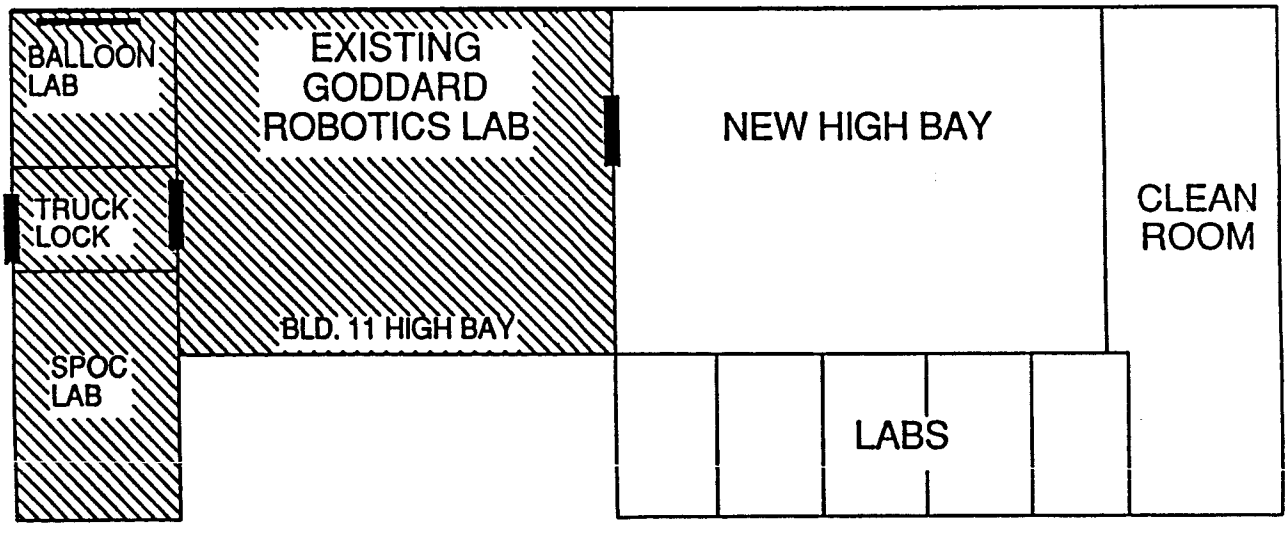


Figure 8. Extension planned to current robotic facility.

5. References

1. Harry Yae, Sung Soo Kim, Edward J. Haug, Warren Seering, Kamala Sundaran, Bruce Thompson, Harold P. Frisch, and Richard Schnurr, "Test and Validation for Robot Arm Dynamics Simulation," *Proceedings of the 1989 NASA Conference on Space Robotics,* Pasadena, CA, January 31-February 2, 1989.

2. Ronald Lumia, "The FTS: From Functional Architecture to Computer Architecture," *Proceedings of the 1989 NASA Conference on Space Robotics,* Pasadena, CA, January 31-February 2, 1989.

3. Gary E. Mosier, Maureen E. O'Brien, and Richard G. Schnurr, "Development of a Robotics Technology Testbed for the Flight Telerobotics Servicer Project," *Proceedings of the 1989 NASA Conference on Space Robotics,* Pasadena, CA, January 31-February 2, 1989.

The Goddard Space Flight Center (GSFC) Robotics Technology Testbed

Rick Schnurr, Lead engineer, GSFC
Maureen O'Brien, GSFC
Sue Cofer, Lead author, Digital Equipment Corporation

ABSTRACT

Much of the technology planned for use in NASA's Flight Telerobotic Servicer (FTS) and the Demonstration Test Flight (DTF) is relatively new and untested. To provide the answers needed to design safe, reliable, and fully functional robotics for flight, NASA/GSFC is developing a robotics technology testbed for research of issues such as zero-g robot control, dual-arm teleoperation, simulations, and hierarchical control using a high-level programming language. The testbed will be used to investigate these high-risk technologies required for the FTS and DTF projects.

The robotics technology testbed is centered around the dual-arm teleoperation of a pair of 7 degree-of-freedom (DOF) manipulators, each with their own 6-DOF mini-master hand controllers. Several levels of safety are implemented using the control processor and a separate watchdog computer, as well as other low-level features. High-speed I/O ports allow the control processor to interface to a simulation workstation: all or part of the testbed hardware can be used in real-time dynamic simulation of the testbed operations, allowing a quick and safe means for testing new control strategies. The NASREM hierarchical control scheme, developed at the National Institute of Standards and Technology (NIST, formerly NBS), is being used as the reference standard for system design. All software developed for the testbed, excluding some of simulation workstation software, is being developed in Ada.

The testbed is being developed in phases. This paper describes the first phase, which is nearing completion, and highlights future developments.

1 Overview of the Robotics Technology Testbed

Much of the technology planned for use in NASA's Flight Telerobotic Servicer (FTS) and the Demonstration Test Flight (DTF) is relatively new and untested. To provide the answers needed to design safe, reliable, and fully functional robotics for flight, NASA/GSFC is developing a robotics technology testbed for research of issues such as zero-g robot control, dual-arm teleoperation, simulations, and hierarchical control using a high-level programming language.

The testbed currently is centered around the dual-arm teleoperation of a pair of 7-DOF manipulators manufactured by Robotics Research Corporation (RRC); see Figure 1. Each arm has an extension of almost 6 feet; they're mounted on one stand approximately 8 feet from the ground, 1.5 feet apart, simulating a right and left arm. Master-slave (teleoperation) control of the RRC arms is accomplished with a pair of 6-DOF mini-master hand controllers from Kraft Telerobotics (Kraft): see Figure 2.

Figure 1: Dual Mounted Robot Research Corporation (RRC) Slave Arms

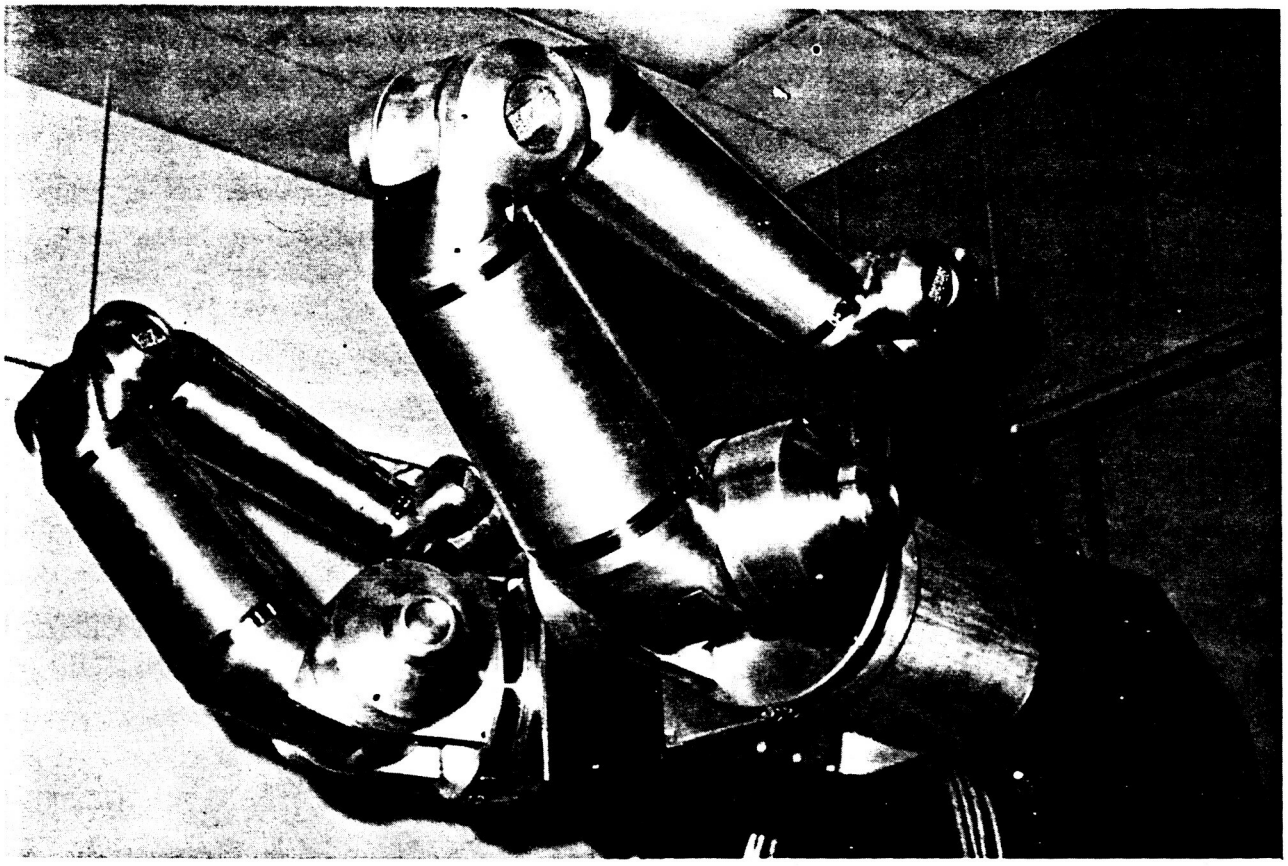
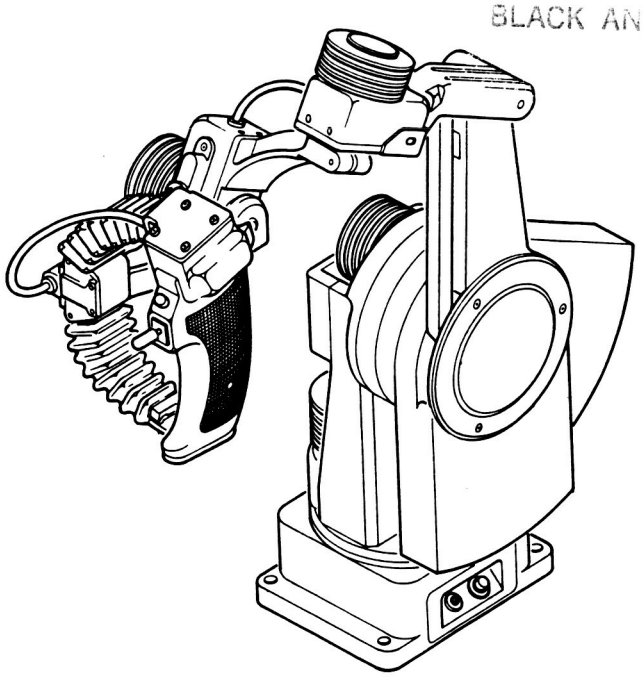
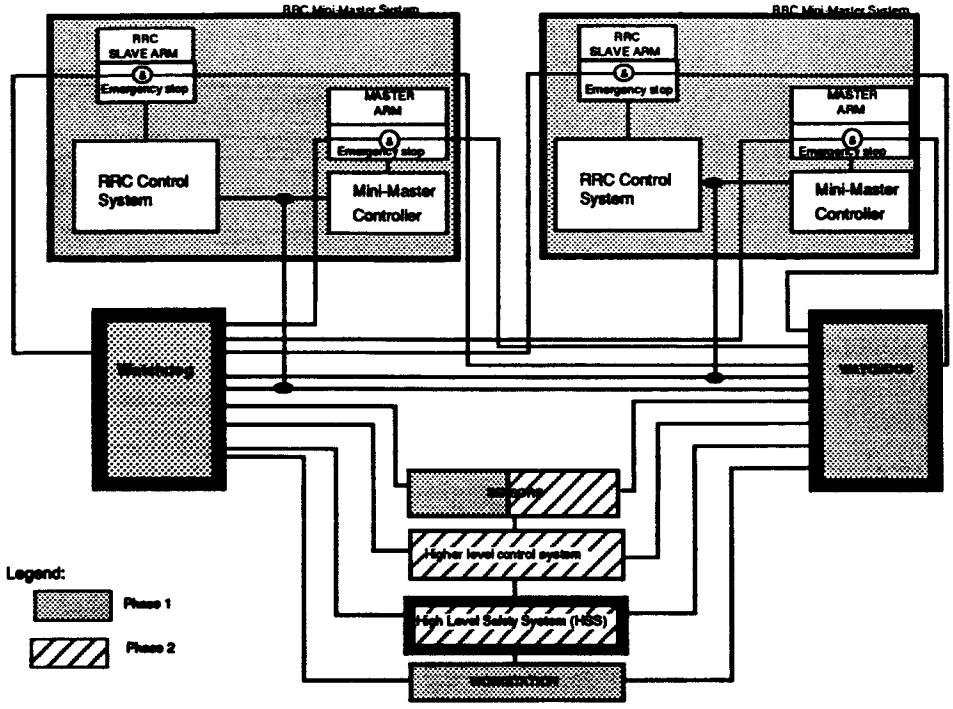


Figure 2: Kraft Telerobotics Mini-Master Arm



Additional control switches on the Kraft master hand controller allow for control of end-effectors mounted of the RRC slave arms. Algorithms implemented in the control processor provide all coordinate transformations between 6-DOF and 7-DOF space as well as several different force-feedback control schemes. Several levels of safety are implemented using the control processor and a separate watchdog computer, as well as other low-level features. Figure 3 diagrams a high-level system integration of the testbed.

Figure 3: Diagram of Robotics Technology Testbed



I/O ports allow the control processor to interface to a Silicon Graphics IRIS workstation. All or part of the testbed hardware can be used in real-time dynamic simulation of testbed operations, including real-time graphics displays of the simulated components. For instance, the system may be configured to allow the teleoperator to drive a computer model of the slave arms using the actual master arms; the operator can watch the slave arms perform the motions in graphics simulation, and force feedback based on the (simulated) dynamics model can be fed back to the control processor of the actual master arms. This approach offers a quick and safe means to test new control strategies.

The NASREM hierarchical control scheme, developed at the National Institute of Standards and Technology (NIST, formerly NBS), is being used as the reference standard for system design. All software developed for the testbed, excluding some of simulation workstation software, is being developed in Ada: this will provide information as to how useful Ada is as a robot control language. Different Ada code designs will be explored, and the performances of each tested for their suitability to robot control.

The testbed is being developed in phases. This paper describes the first phase, which is nearing completion, and outlines the remaining phases. Phase I includes set up of the physical testbed, design and manufacture of required hardware for system modifications and interfacing, design and implementation of RRC control modifications, derivation of optimal jacobians and kinematic matrices for both the master and slave arms, and design and implementation of a watchdog (servo level) safety system for testbed operations. At the end of Phase I many of the hardware components for the testbed will have been installed and tested. A basic robot control algorithm will have been implemented, and the safety system will be completely defined, the required hardware procured hardware, and the initial software integrated into the RRC robot system.

2 Kinematics

The forward kinematic matrices, referenced in this paper, are 4x4 homogeneous transformations which use joint angle data and relates the base coordinate frame to the end-effector coordinate frame; given joint angles and the base coordinate frame, then, the pose (position and orientation) of the end-effector can be determined. Similarly, inverse kinematics provide a (complex) relationship from a position in the end-effector coordinate frame to the manipulator joint angles. The jacobian matrix specifies a mapping from joint anglular velocities to resolved cartesian velocities in the end-effector coordinate frame. All kinematic and jacobian matrices used in the testbed contol system, unless otherwise noted, are generated using the following procedure:

- Generate the required input equations for MACSyma: MACSyma is a Lisp-based system written by Symbolics that performs symbolic algebra.
- MACSyma is executed; the output is actual FORTRAN code of the matrix.
- The FORTRAN code is downloaded to an IBM-PC.
- An optimizer has been written by the robotic technology testbed team to optimize the MACSyma generated code; it runs on the PC, using the machine-generated FORTRAN code as input. The optimizer extracts all occurrences of trigonometric functions and assigns their value to a variable, which will be used in the actual matrix computations: this ensures all trigonometric functions are computed only once. Next, in a recursive algorithm, common factors in the MACSyma equations are pulled out and, again, their value assign to a variable which will be used in actual matrix computations. Lastly, the optimized code is translated to Microsoft C-language; the optimizer is being extended to generate optimized Ada code.
- The optimized C code is then tested using numeric examples generated by MACSyma.

The timings presented in this paper, then, represent the execution speed of the resulting optimized optimized code in Microsoft C on a Compaq 386/v20.

2.1 The Kraft Mini-master

The Kraft mini-master forward kinematics matrix are fairly straightforward.

The Kraft jacobian transpose is a 6x6 matrix which relates the forces seen at the Kraft handle in handle coordinates to Kraft joint torques. This matrix can be used to reproduce forces/torques at the handle of the mini-master which were read by a wrist sensor on the RRC slave arm. The execution speed of the resulting optimized code is 1.7 msec.

The Kraft jacobian transpose multiplied by a 6 element force vector produces a matrix which directly relates Kraft handle forces to Kraft joint forces. This matrix was computed and optimized separate from the jacobians above in order to minimize the number of terms which needed to be computed. The execution speed of the resulting optimized code is 1.8 msec.

A dynamic model for the Kraft is in process.

2.2 The RRC Slave Arms

The RRC forward kinematics matrix is the homogeneous transformation (position/orientation) between the RRC base coordinates and the RRC end-effector coordinates.

The RRC jacobian transpose is a 6x7 matrix which relates the forces seen at the end effector in end-effector coordinates to the joint torques. This matrix can be used for impedance or compliance control. The execution speed of the resulting optimized code is 1.8 msec.

The RRC jacobian pseudo-inverse is a 6x7 matrix which relates a cartesian velocity vector measured in the end-effector coordinates of the RRC to a joint space velocity vector for the RRC. Since the RRC is a redundant manipulator (7-DOF) there are an infinite number of possible inverse jacobian matrices. The matrix used in the testbed control system is based on the following Moore-Penrose pseudo-inverse equation:

$$J^{dagger} = J^T * (J * J^T)^{-1}$$

where J^{dagger} is the jacobian psuedo-inverse. The derivation of the equation is based on minimizing the sum of the squares of the joint velocities; this maximizes the speed of the arm. The disadvantage to this is that the elbow (the redundant motion plane) is allowed to move without restraint.

The $J * J^T$ matrix has been derived using the process outlined above (optimized MACSyma code); software, based on Gaussian elimination, had to be written to numerically compute the inverse of this matrix because it is too complicated for MACSyma to invert symbolically. A breakdown of the timings for the resulting optimized code are:

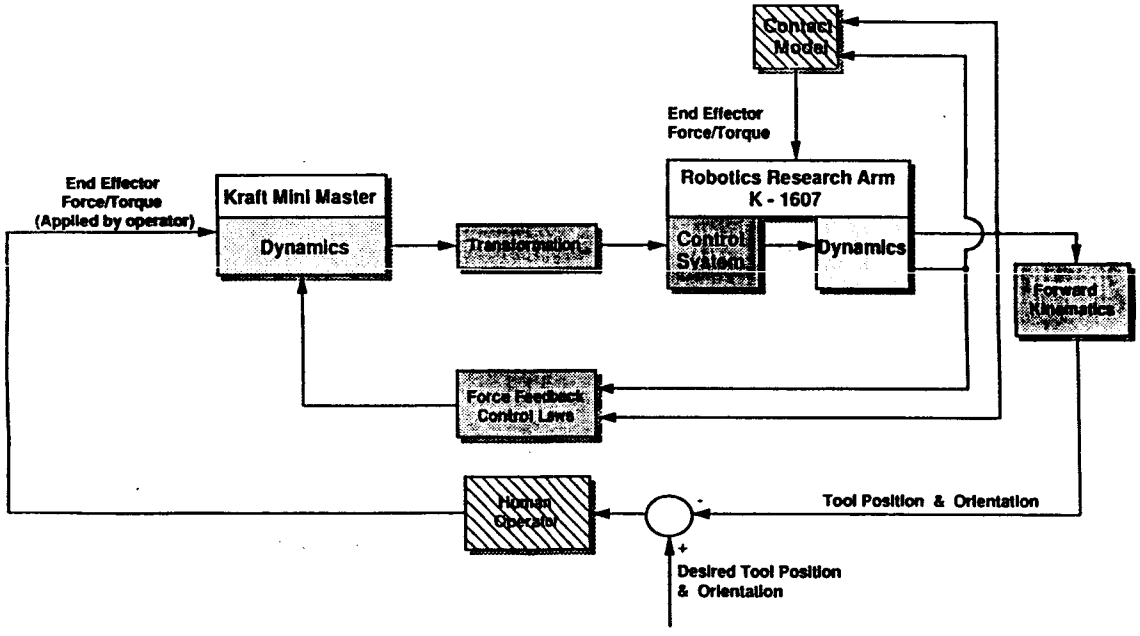
1.8 msec - compute the jacobian J
2.7 msec - compute $J * J^T$
9.7 msec - invert $J * J^T$ and then multiply it by J^T
14.2 msec - total execution time to compute the jacobian psuedo-inverse.

Other inverse kinematics schemes have been explored; at least one of them will be implemented and tested, in addition to the jacobian psuedo-inverse described above, during the Phase I effort.

3 Control Algorithms

Phase I of the GSFC robotic technology testbed is concentrating on the NASREM servo level control. Trade-off analysis between using the joint torque sensors and a wrist force torque sensor are continuing; both will probably be implemented. A high-level diagram for the testbed teleoperation control loop is shown in Figure 4.

Figure 4: GSFC Robotic Technology Testbed Teleoperation Control



3.1 Force Reflection

The first attempt at force reflection will be done in cartesian coordinates. A wrist sensor mounted on the RRC slave arm(s) is used as the source of force/torque data; these forces/torques will then be translated back to the Kraft mini-master. There are two types of force feedback which have been explored: position-position and force-rate. The position-position force reflection generates a torque on both the master and slave proportional to the differential position between the two. When the rigidity of the RRC control loop and the weight of the manipulator are considered, position-position force feedback does not look promising. However, due to the large size of the RRC slave arms, they

cannot move fast enough to keep up with the smaller mini-masters; some means must be found to keep the operator of the mini-master from getting to far ahead of the slave manipulator. A variant of position-position force reflection which matches the master arm motion to the slave arm motion accomplishes this goal.

The force-rate mode of force reflection takes data from a force/torque sensor on the slave and uses this data to generate force/torque commands on the master. The difference in position between the master and slave causes a torque to be impressed on the slave. The differential rate generates torques on both the master and slave.

A fully robust force reflection algorithm will ultimately integrate both control methodologies: a design and implementation for this integration is currently in progress.

3.2 Indexing

Another major subject which must be addressed is the indexing scheme. Indexing is the act of disabling the control algorithm for the master arm, allowing the operator to move the master to a more comfortable (workable) position, and then re-initializing the control parameters and resuming teleoperation of the slave arm. There are two challenges embedded in the indexing problem: first, to re-establish the control link between the master and the slave without moving the slave. Second, to resolve the reflected forces on the indexed master so the axis of motion match the force reflection axis of the slave arm. Separate matrix formulations for indexing will be required, and are currently being calculated and optimized using the method outlined in Section 2.

Experimentation with one indexing scheme has been completed: however, after analysis of the positioning requirements, one indexing scheme is not going to satisfy all possible teleoperation tasks and a hybrid of different control schemes will ultimately be required. This issue will probably not be totally resolved until later phase developments.

3.3 Teleoperation

Two formal approaches to teleoperator control will be explored (see Section 2). Both center on developing formal control equations, performing stability analysis/simulations, simulating control algorithms using dynamic simulations, and then implementing the control code in Ada. One group will be focusing on advanced adaptive control schemes: this group is funded through a grant to Catholic University. The second is the testbed team, which will be heavily involved in development of control algorithms using the real robots and DTF and FTS tasks as a basis. These efforts are scheduled to be completed next year.

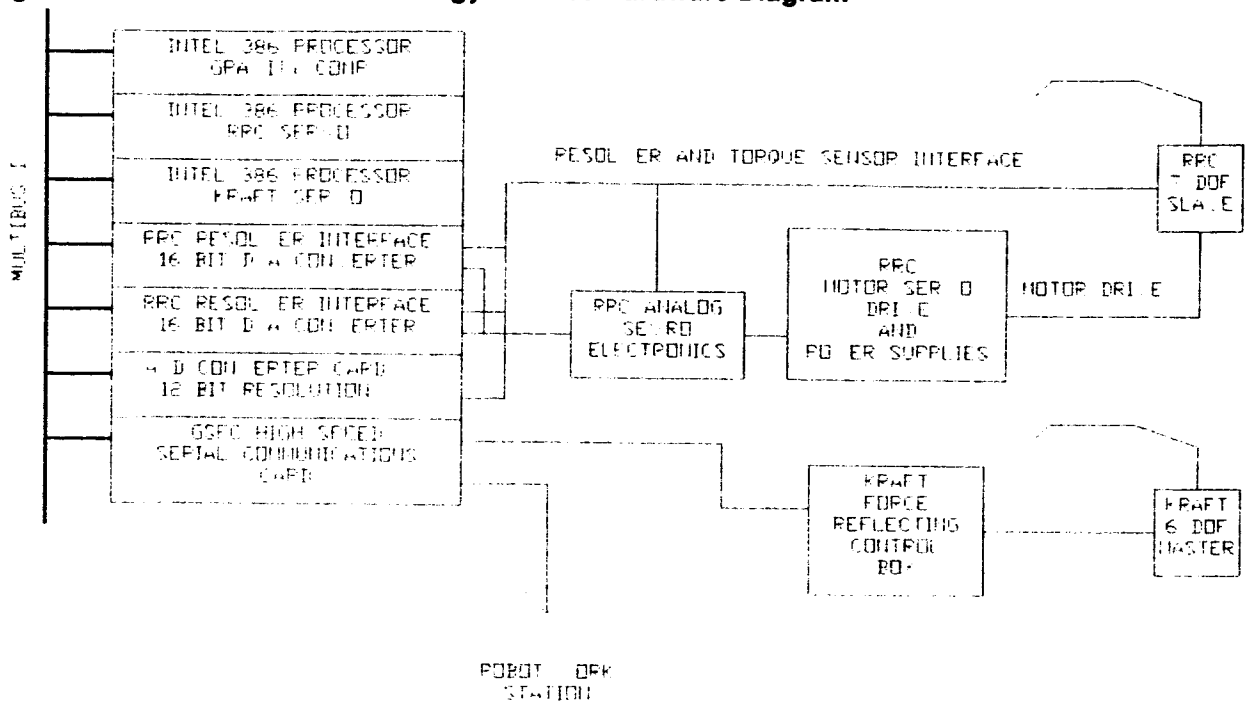
3.4 Data Collection and Testing

Data collection and testing will ultimately be required for dynamic simulation efforts. All tap points available in the RRC controller have been brought out to connectors where they can be hooked to a oscilloscope and later to a data acquisition system. One possible data acquisition system is the Safety system. The Safety system, by design, already requires most of this data to perform it's watchdog functions (see Section 6). In fact, once the Safety system is interfaced with a micro-VAX a complete self contained data collection/reduction station will exist.

4 Hardware

The GSFC robotic technology testbed control electronics are shown in Figure 5.

Figure 5: GSFC Robotic Technology Testbed Hardware Diagram



The testbed control computer will use three Intel 80386 processor boards, two RRC resolver (analog) to digital/digital to analog converter (DAC) cards, one Intel parallel I/O card, a 12 bit resolution analog-to-digital converter (ADC), and one high speed serial card, all on a multibus I.

The first processor, the RRC servo, will read resolver position, the analog joint velocity, and the joint torque sensor for each joint of the RRC arm. It will then calculate a motor torque and set a DAC on the resolver card.

The second 80386 processor, the Kraft servo, will be interfaced to a Kraft mini-master over a RS422 link. This processor will read position data from the mini-master, implement the force reflecting control algorithm, and send joint torque data to the mini-master.

The third 80386 processor, gravity compensation, serves two functions: first, it computes the torques which will be seen at each slave joint due to gravity. To do this the computer will have to execute a recursive force transformation algorithm, requiring the positions and masses of the link centers of gravity. The expected joint torques will be available to the other processors for use in compensating for gravity. The second function performed by the processor is to provide a way to limit allowable end-effector positions and to check for run away controller behavior: if either is detected operations of the robots will be shut down.

The RRC resolver interface cards allow the testbed control system to interface directly with the RRC joint resolver and torque sensor. It receives joint resolver and torque data from the the RRC arms and makes it available to the testbed control system. There are also DACs present on the card: these convert the motor torques, calculated by the testbed control system, to analog signals and sends them to the existing RRC analog servo electronics.

The analog-to-digital converter (ADC) card will be used to digitize the velocity and torque signals with 12-bit accuracy. The signals read by the ADC are buffered by a GSFC designed buffer card. This card makes these signals available to the Safety system without allowing the Safety system to corrupt them.

Seven analog boards (not shown in Figure 5) are connected to the existing RRC motor servo drive cards. These analog boards serve a dual function: first, they process the motor voltage and motor current signals for the Safety system, and, second, they compute the motor rpm. The motor rpm data is compared to the resolver velocity; also, the joint velocity is compared to a preset velocity limit. A deviance in either will cause the RRC robot to shut down. The velocity limits will be set very low when new servo control software is being tested.

The digital I/O card (not shown in Figure 5) is the computer's interface with the RRC joint home switches, the servo/enable status indicators, and the output drivers for panel lights and the arm enable relay.

The high-speed serial interface subsystem can be broken down into two parts: the first communicates with the Kraft mini-masters over a 2-wire packet interface, RS422 asyncronous protocol at 93750 baud. The Kraft is a slave in the communications protocol; when a valid torque command packet is received by the Kraft controller, a position report packet is sent out. The second part of the serial interface communicates with the Safety system and the RRC workstation. These interfaces are still in their definition phase. Long term it would be desirable to use a standard networking protocol between these elements. Candidates would be IEEE 802.3, MIL-STD-1553B, or high-speed serial multi-drop.

4.1 Modifications to the Existing RRC Arms

Two 80386 processors were added to the 80386 processor already present in the controller. A 12-bit ADC was added to measure joint torque and velocity signals. A buffer board was added to isolate critical signals for safety system. Seven analog boards were added to buffer and filter motor voltage and current. These motor interface boards also calculate motor velocity using motor voltage, motor current, and motor circuit resistance.

The testbed RRC joint control loop, which incorporates the new hardware, is as follows: the ADC card will be used to digitize the velocity and torque signals read from the RRC robot joints with 12-bit accuracy. The signals read by the ADC are buffered by a GSFC designed buffer card. This data is used by the first 80386 processor to compute the desired joint motor torques, which is sent to a DAC on the resolver card. The voltage from this DAC chip is connected to the analog torque loop electronics. The existing motor servo amps for the RRC arms can then be driven by the output of this motor torque loop.

The high-speed serial card has been installed and is working in RS232 mode. The PC board is currently being modified to interface with the RRC high-speed serial card in RS422 mode at 288K baud.

5 RRC Arm End-effectors

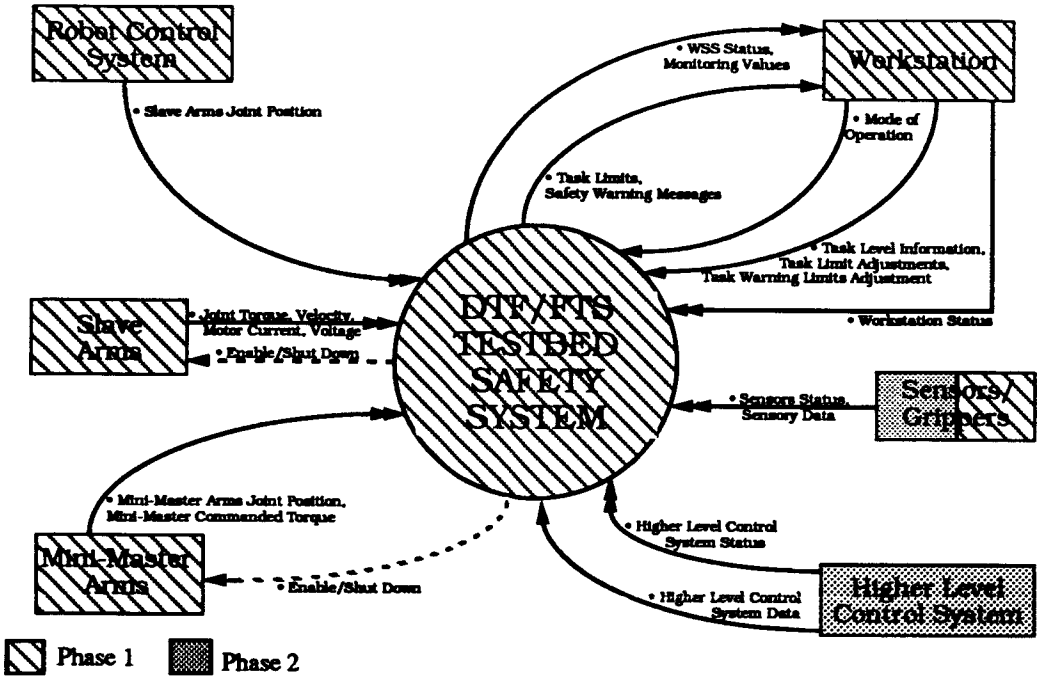
Several end-effectors, and a tool change and storage concept are currently being designed for the RRC arms. There will be interface requirements between the end-effector controller and the robot control at two levels. The first allows the end-effector to operate. Examples of this are using tool plate roll to unscrew a hex bolt through a ratchet, using tool plate roll along with a special fixture to install and stow end-effectors, and using the robot's wrist and joint force/torque sensors to provide active compliance. The second level allows the robot and the operator to verify proper execution of robot related tasks. Examples of actions which need verification are gripping something and checking that gripper distance is correct, verifying the proper tool is attached to the robot, verifying status and actions of tool change out apparatus, and verifying and monitoring proper gripper operation.

A final design concept for the interactions between the RRC controller and the controller for the end-effector(s) will exist at the end of Phase I. Also, several prototype controllers will have been tested on PUMA robots. The construction of the computer based end-effector controller(s) and the software required to run it will be completed. Mechanisms to give the capability of having a runoff between several different ORU change out tools will have been built, and a tool auto-change and storage unit will have been designed and be near the end of it's fabrication cycle by the end of Phase I.

6 The Safety System

Figure 6 diagrams the data flow between the safety system and the other components of the technology testbed telerobot control system.

Figure 6: Safety System Diagram



The Safety System is designed to ensure the safe operations of the RRC and Kraft mini-master arms. Safe operations, as defined by the testbed team, requires that operation of the robot does no damage or harm to an operator or bystander, to itself, to any objects in the robots workspace. Ironically, this is a very unexplored area of robotics; as a result, the robotic technology testbed has turned out to be a testbed for exploring robot safety technology as well.

The current testbed Safety System is composed of several subsystems. The High-level Safety System (HSS) has knowledge about the task that is being performed and determines the safe operational and warning limits for the task. These limits are sent to the Watchdog Safety System (WSS). The WSS, which exists at the servo level of telerobot control, monitors robot and sensor data to ensure that the data is within the safe limits determined by the HSS. It is also responsible for monitoring the health of other computers in the testbed robot control system such as the workstation computer and the robot controller.

6.1 The Watchdog Safety System (WSS)

The Watchdog Safety System monitors the health status of the testbed control system, monitors robot and sensor data to ensure that the robots are operating within safe limits, and safely shuts down the robot(s) when an unsafe condition exists.

There are many functional requirements for the WSS. It must be two fault tolerant, or redundant to fail to safe when certain hardware errors are detected. The WSS monitors the health status of different subsystems of the testbed telerobotic control system; if any of the components fails, WSS shall safely shut down the robot. The WSS also monitors the robot operational data to ensure it is operating within safe limits. Operational data includes motor current, joint position, joint velocity, joint acceleration, joint torque, sensor data, and positions entering forbidden volume. The WSS also monitors the state of the robot during operations, ensuring, for example, that end-of-arm tooling and workpieces are not inadvertantly released (in zero-g, this condition could be disastrous).

The WSS, Figure 6, was designed to meet these functional requirements. The WSS receives the required safe operational limits, listed above, from the HSS; the workstation operator can override these limits to a more constrained boundary, but cannot increase them. All subsystems must transmit health status data at regular (to be defined) intervals. The WSS also transmits system status to the workstation and HSS subsystems.

There are many safety issues which are not clearly defined; for example, what is meant by "safe shut down of the robot". This is usually defined as immediate cessation of robot motion; however, nothing is said about the state in which the robot is left. Shutting down the robot could involve simply applying brakes immediately to all robot joints, backing off (reflex withdrawal) and then applying brakes instantly applying brakes and then putting the robot to a compliant mode, or simply stop sending signals to the robot, leaving the robot in the last known safe state. There are safe and unsafe aspects to all of these approaches, usually dependent on the particular robot task in which the anomaly occurred. One of the things the testbed safety team will be looking into is analyzing the impact of the different safe shut down schemes on both the system and the operating environment.

Directly related to the issue of a safe shut down is the definition of a safe return to operation after a shut down has occurred. A specific restart procedure is not clearly defined: it could involve recalibrating the robot, returning to home, resetting operational parameters, or specific operator action(s) could be required, to name a few. Again, this is one area to be researched by the testbed safety team.

6.2 The High-level Safety System (HSS)

This system uses task level information to determine operational and warning limits for the WSS. It is suspected that some level of colision avoidance will be performed at this level. The HSS will be completely defined at the completion of Phase I.

7 Future Development of the Robotics Technology Testbed

The following goals are proposed for the Phase II effort of the robotics technology testbed implementation:

- Implement improved force reflecting algorithms.
- Incorporate the higher levels of the NASREM model, written in Ada, into the RRC control system for autonomous operations. This system, the Hierarchical Ada-language Robot Programming System (HARPS), is currently under development at the GSFC robotic technology testbed: Stephen Leake of NIST is the lead engineer. A paper describing HARPS is being presented and published at the 1989 IEEE International Conference on Robotics and Automation in May, 1989.
- Integrate end-effector controller(s) into existing RRC control, generate tasks to exercise it, and perform tasks. Data will be collected from these tests and presented as a deliverable item. The task set will include ORU latching and unlatching, truss node assembly, and robot to robot hand off of object(s).
- Support the dynamic simulation parameter characterization effort.
- Demonstrate the validity of the University of Iowa's dynamic model using actual RRC robot data. The University of Iowa will prepare a plan involving RRC robot tests and perform these tests under the direction of testbed team members. The reduction of data from these tests may lead to a better model of the RRC robots.
- Demonstrate validity of the University of Iowa's IRIS-IRIS model using actual RRC robot data. Two independent IRIS systems, interfaced over an ethernet, are used for the model: one to generate data and the other for graphics. The model validation will be done at the same time the other Iowa model is verified.
- Investigate sensor technologies, specifically how beneficial different sensor data would be to the Safety System.

At the end of Phase II the entire telerobot system will be fully integrated and tasks will have been performed to demonstrate its capabilities. Phase II is scheduled to be completed in August of 1989. A detailed definition of future phases will be completed during the Phase I effort.

TEST AND VALIDATION FOR ROBOT ARM CONTROL DYNAMICS SIMULATION

K. Harold Yae, Sung-Soo Kim, Edward J. Haug
The University of Iowa

Warren Seering, Kamala Sundaram, Bruce Thompson
Massachusetts Institute of Technology

James Turner, Hon Chun
Cambridge Research

Harold P. Frisch, Richard Schnurr
Goddard Space Flight Center

Abstract

The Flight Telerobotic Servicer (FTS) program will require an ability to develop, in a cost effective manner, many simulation models for design, analysis, performance evaluation, and crew training. Computational speed and the degree of modeling fidelity associated with each simulation must be commensurate with problem objectives. To demonstrate evolving state-of-the-art general-purpose multibody modeling capabilities, to validate these by laboratory testing, and to expose their modeling shortcomings, two focus problems at the opposite ends of the simulation spectrum have been defined:

(1) *Coarse Acquisition Control Dynamics*
Create a real-time man-in-the-control-loop simulator. Provide animated graphical display of robot arm dynamics and tactile feedback sufficient for cueing the operator. Interface simulator software with human-operated tactile feedback controller; i.e., the Kraft mini-master.

(2) *Fine, Precision Mode Control Dynamics*
Create a high-speed, high-fidelity simulation model for the design, analysis, and performance evaluation of autonomous 7 degree-of-freedom (dof) trajectory control algorithms. This model must contain detail dynamic models for all significant dynamics elements within the robot arm, such as joint drive mechanisms.

Successful completion of this project will require the cooperative efforts of several research groups, each focusing within a prime area of responsibility and jointly working within an interface area. Our intent is to utilize the recently developed recursive multibody dynamics algorithm associated with Order N Iowa, to create a real-time man-in-the-loop simulator for the Robotics Research Corporation (RRC) 7 dof robot arm in the Goddard Space Flight Center (GSFC) robotics laboratory. Man-in-the-control-loop will be via a fully interfaced Kraft mini-master tactile feedback controller.

We further intend to transport the recursive multibody dynamics equations to old DISCOS to create Order N DISCOS, a new high-speed, high-fidelity general-purpose control analysis capability. Pilot demonstration of Order N DISCOS will be via application to the precision control modeling needs associated with supporting RRC 7 dof robot arm autonomous controls design and analysis. Fine detail modeling will require detailed power train modeling in a format compatible with definition within Order N DISCOS. A series of control algorithms and associated sets of laboratory tests will be defined. These will be performed at GSFC and used to validate our ability to develop a broad range of high-speed, high-fidelity simulation capabilities in a cost effective manner.

1. Introduction

Recent advances in high-speed parallel processing computers, and new methods in dynamics formulation that exploit modern computer architectures have created a substantial increase in computational speed; consequently, dynamics simulation is, in some cases, even faster than real-time. Also, high-speed computer graphics generates high-fidelity animation of the simulation, and thereby creates realism sophisticated enough to give an adequate visual cue to the operator.

Here we demonstrate the use of such advanced technology; specifically, we develop, in a cost effective manner, simulation capabilities of the RRC 7 dof arm for design, analysis, performance evaluation, and crew training in support of the FTS program. The simulation model is to be validated with a series of experiments in GSFC to ensure that it represents the actual model to the highest degree of fidelity possible. Once validated, the simulation model is to be tied with high-speed computer graphics and the Kraft mini-master to give the operator visual and tactile feedbacks.

2. Development of Order N Iowa

Dynamics analysis of multibody mechanical systems requires formulation of the equations of motion in a differential equation form and associated constraints as nonlinear algebraic equations. In deriving the equations of motion, two basically different kinds of generalized coordinates are used; one is joint or relative coordinates between two contiguous bodies, the other is Cartesian or absolute coordinates of each body. The Cartesian coordinate formulation is quite general and treats open- and closed-loop mechanisms in the same way, but at the same time it introduces a maximum number of generalized coordinates and associated kinematic constraints. On the other hand, the joint coordinate formulation employs a minimum number of generalized coordinates and is directly applicable to open-loop mechanisms, but it requires some extension to treat closed-loop mechanisms.

In the recursive formulation [1,2], dynamics analysis can be divided into three major steps. First, by using the variational-vector calculus approach [3], the variational equations of motion are formed in Cartesian coordinates. At this stage, the known positions and velocities in either Cartesian or joint coordinates must all be expressed in Cartesian coordinates. For example, in the case of a robot arm, the base body is described with respect to the inertial frame, but the others may be described in relation to their neighboring members, namely, in joint coordinates such as joint angles and joint angular velocities. Then by starting from the base body and proceeding toward the tree-limb-end body the joint coordinate representation can be transformed to the Cartesian coordinate representation. Second, the variational equations of motion in Cartesian coordinates are transformed into the variational equations of motion in joint coordinates by recursive use of the kinematic relationship between two contiguous bodies. Third, the equations of motion are finally expressed in joint coordinates. From the equations of motion acceleration is found; then, through numerical integrations, velocity and position are found. This concludes one cycle of iteration.

The recursive formulation has been applied to a variety of mechanisms, and has successfully demonstrated its efficiency [1,2]. Furthermore, it is easily adaptable to the emerging parallel processing computers.

3. Development of Order N DISCOS

The DISCOS multibody dynamics software was originally developed for the Goddard Spaceflight Center during the mid-1970's for analyzing the response of a spacecraft that could be modeled as a collection of rigid and flexible bodies. Small displacement structural flexibility could be handled by allowing the spacecraft to be modeled by a general-purpose finite element code such as NASTRAN. By modeling individual bodies, rather than entire structures, the overall vehicle can experience both large motions relative to inertial space as well as large motions between individual bodies, without having to compute new structural parameters for each possible configuration. The basic DISCOS methodology makes use of advanced analytical dynamics formulation techniques that model individual bodies of the system and impose kinematic constraint conditions to force the correct overall system-level dynamical response.

The key to success in this approach is the use of the Lagrange multiplier technique in order to enforce the interconnection topology. This process successfully overcame several of the multibody formulation problems that had plagued earlier efforts at obtaining general-purpose software. The basic algorithm requires that the system-level Lagrange multiplier be computed during each integration step. The solution for the Lagrange multiplier is defined by a simple linear algebraic matrix equation whose dimension is governed by the number of constraint conditions which exist between contiguous bodies. Since the number of constraint conditions tends to increase more rapidly than the number of bodies, the calculation of the Lagrange multiplier linear matrix equation effectively limits the practical upper limit for the number of bodies which can be simulated. Although the exact number is somewhat problem-dependent, typical simulation runs with more than twelve bodies are not common.

As currently implemented, the DISCOS algorithm is now described as an order N^3 process, where N is the number of bodies in the multibody simulation. Clearly, as N increases, the computational burden is increasing at a significant rate, and real-time applications are not a practical reality. To support emerging needs for real-time autonomous robotics applications on the proposed space station, the current version of DISCOS is being upgraded to

incorporate recently developed order N recursive multibody formulations. This upgrade, by Cambridge Research, will occur over a three-year period and is being carried out as part of the Industry/University Cooperative Research Center (I/UCRC) for Simulation and Design Optimization of Mechanical Systems at The University of Iowa. Order N algorithms allow the analyst to integrate the minimum dimension set of equations at the acceleration level. For a tree-like structure, the Lagrange multiplier calculations completely disappear from the calculations, though they can be produced if there is interest in loads information at joints. For ring-like structures, however, Lagrange multipliers are still required in order to deal with closed-loop systems. However, because the dimension of the Lagrange multiplier required is limited to the constraint conditions applied at a single hinge, the calculations are greatly simplified. Another significant advantage that the order N algorithms have over conventional order N^3 algorithms is that the basic computational structure is readily applicable to parallel implementations, as demonstrated in the pioneering work by the Iowa group. By combining both the recursive character and the ease of parallel implementation of order N algorithm, the proposed upgrade of DISCOS furnishes an enabling technology development for real-time on-orbit space station robotics activities.

Other planned enhancements for the DISCOS software include upgrades for event-driven activities such as (i) intermittent kinematic constraints (e.g., inequality constraint), (ii) intermittent loop closure (e.g., variable ring/tree topology for transitions between get and move and transitions between move and put operations for robots), (iii) multi-arm robot payload handoff (e.g., variable tree topology), (iv) constraint stabilization and momentum balance methods, and (v) differential/algebraic equation solution methods.

All planned upgrades of the DISCOS software are to be made so as to preserve the input/output characteristics of the existing software and to minimally impact the existing DISCOS user group. The evolving software capabilities will be validated with ground-based robotics tests at the Goddard Spaceflight Center during the summer of 1989 as well as being compared with the Order N Iowa software being developed at The University of Iowa

4. Dynamics Modeling and Simulation

As a generic model for space teleoperation, the RRC robot arm is simulated to support real-time man-in-the-loop control. The RRC robot arm has seven relatively rigid link segments connected at revolute joints [4]; each segment is a thin-wall exoskeletal structure. All the joints are directly driven by drive actuators directly mounted at the joints.

For dynamics modeling, the body reference frames are defined as in Figure 1, where all X-axes are defined along the joint rotational axes. The origin of each body reference frame is at the center of gravity of that body. Bodies 1 to 7 are identified as shoulder roll, shoulder pitch, elbow roll, elbow pitch, wrist roll, wrist pitch, and tool-plate roll, in that order. In addition, the home configuration is defined as follows: the roll axes of bodies 1, 3, 5 and 7 are on the same vertical plane; the roll axis of body 3 makes 60° with the roll axis of body 1; the roll axis of body 5 makes 30° with the roll axis of body 1; the roll axis of body 7 is perpendicular to the roll axis of body 1; all the initial joint angles are set to zero in that configuration.

The robot arm dynamics model has been created with the recursive formulation and simulated with parallel computation on an Alliant FX/8 multi-processor mini supercomputer

To estimate computation time for this model without any joint actuator, a free fall motion under gravity is simulated using different numbers of processors. Here the numerical integration is done by the Adams-Bashforth third-order method with a 10 milisecond constant step size. In Figure 2, the computation time with 4 processors is 4.35 miliseconds per time step, which means that it takes 0.435 second for 1 second real-clock time simulation. Furthermore, with 8 processors the computation time is only 2.77 miliseconds per time step; in other words, the simulation is 3.5 times faster than the real-clock time. The result of this simulation strongly indicates that the real-time man-in-the-loop simulation is feasible.

5. Control Algorithm Design

The MIT group is currently developing a model of the control system for the RRC robot arm. The configuration of the controller is the same for each of the seven joints, and consists of a velocity and torque compensator. The MIT model will include the effects of the electronics, amplifier, motor and harmonic drive in each joint, since these components are considered important in obtaining an accurate model. The current model takes into account the stiffness in the harmonic drives and viscous friction in the motors, harmonic drives, and links. The next

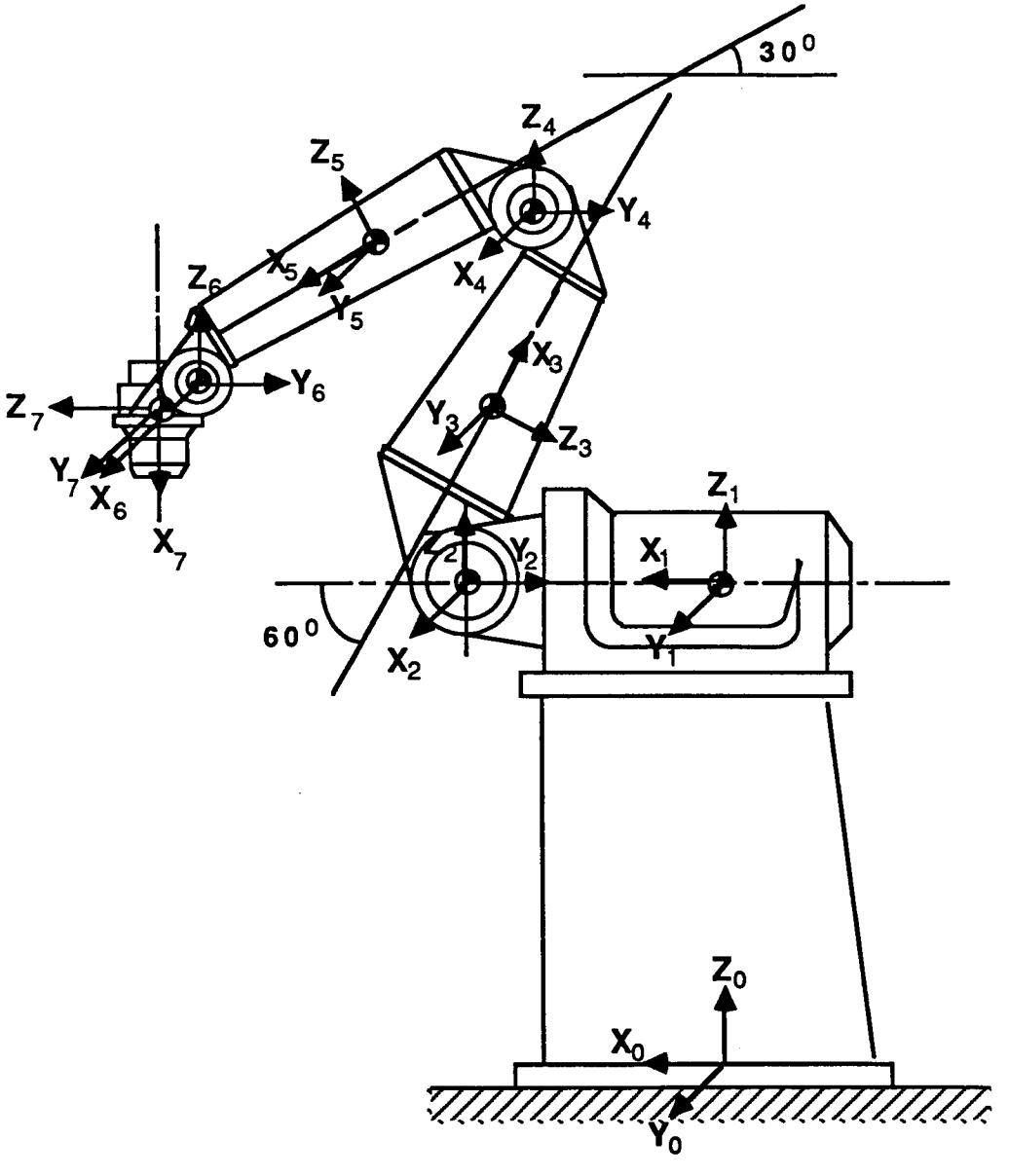


Figure 1. Body Frame Definition and Home Configuration

version of the model will include the effects of the motors and the nonlinear spring characteristics of the harmonic drive. Once this model is finished, it will be combined with the arm dynamics model being developed by Cambridge Research. The complete system model will be verified in both the time and frequency domains using results obtained by GSFC. If the model does not accurately predict the response of the real arm, the modeling assumptions will be re-evaluated and the model will be subsequently updated.

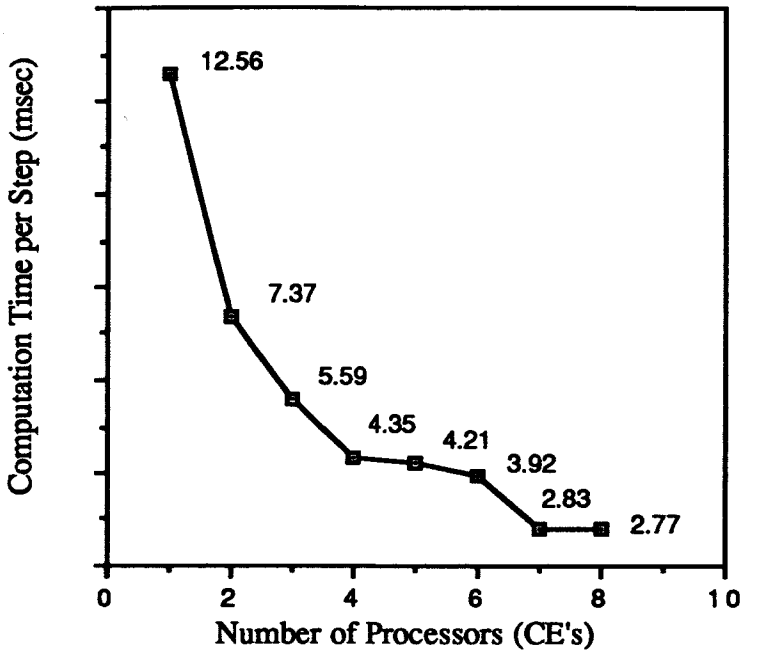


Figure 2. Computation Time on the Alliant

The experimentally verified model will be a very powerful tool for the analysis of new control algorithms. It will allow a researcher to make changes to the system and evaluate their effects without adjusting or replacing any hardware. This will be very useful when different control schemes are under consideration. Areas of future investigation include force control, adaptive control, and reduction of system vibration.

The choice of a particular control scheme is highly dependent on the nature of the task that the robot is to perform. For applications where the robot must interact with its environment, force control provides advantages over conventional trajectory control. The MIT group will examine the feasibility of such a scheme for the FTS. In situations where the model can not be determined accurately, adaptive control may provide an alternative. For example, the MIT group could develop a controller that generates force commands and then corrects for nonlinearities on the basis of the actual force measured at the end effector. For a system with flexibility in the robot or payload, vibration at the end effector could affect robot performance. If this is the case with the FTS, a technique recently developed at MIT can be applied for preshaping the command inputs to reduce vibration significantly.

6. Test and Validation

The controller at each joint of the RRC arm is built on the basis of an approximated linear model; therefore, its performance should be first tested within the vicinity of a certain configuration, where the load and arm inertia is nearly constant. In such a case, a small step or ramp input drives one joint while all the other joints are locked. The same experiment is to be performed throughout all the seven joints. Next, a large step or ramp input drives one joint at a time with all the others locked. This time, nonlinear dynamics will affect the performance of the joint controller, and will also influence the corresponding simulation.

Once we have confidence in the model and controller, more than one joint are to be activated and the robot arm thus maneuvers along a predetermined trajectory in space. During such a maneuver it is necessary that signals pertaining to applied torque and angular displacement and velocity be recorded at each joint for the verification of a dynamics model. Those two types of information, namely, torque and configuration, can uniquely determine the dynamics of the arm and reproduce the same dynamic behavior. In addition, for the verification of a control

algorithm, it is also necessary that at each joint the command reference angle be recorded with the armature current in a servomotor. In other words, the analog signals from the resolver, the torque transducer, and the armature current at every joint should be digitized and recorded, so that each experiment produces two sets of data in the time domain: torque and configuration.

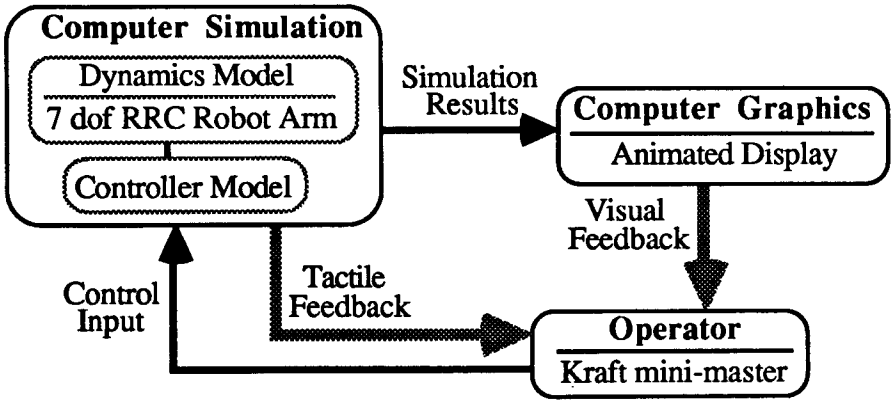
As a first step toward a correct dynamics model, the robot arm and the associated controller are to be simulated and compared using both Order N Iowa and Order N DISCOS. Such a comparison should help to eliminate errors in the simulation model.

Next, simulation results are to be compared with actual experimental data in terms of configuration and torque histories. An important task in such a comparison is first to match the initial configurations between simulation and experiment. Considering the fact that the dynamics of a robot arm can be completely described by configuration and applied torque, the experimental time histories of configuration and applied torque should be applied to the simulation model one at a time to examine how close the model is to the actual system.

First, when an experimental torque history is fed into the simulation model, the forward dynamics analysis produces the corresponding configuration history; that is, angular displacement, velocity, and acceleration in the time domain. Thus the two configuration histories, one from experiment and the other from simulation, can be compared. Next, an experimental configuration history is fed into the simulation model; then, the inverse dynamics analysis produces the corresponding torque history that would have caused the configuration history. This time these two torque histories can be compared.

7. Real-time Man-in-the-Loop Simulation

The operator, as shown in the diagram, is the decision maker in the control loop with visual and tactile feedbacks: visual feedback from computer graphics display of the RRC robot arm, tactile feedback from the Kraft mini-master that is interfaced through a serial port with the graphics workstation. His control action drives the dynamics and control simulation, which is carried out on a high-speed parallel processing system, such as an Alliant FX/8 mini supercomputer, to achieve real-time performance. The result from the simulation is sent to the graphics workstation via the Network Computing System (developed by Apollo Computer, Inc.) and is animated. At the same time, it is also sent to the Kraft mini-master to give tactile feedback to the operator.



For the animated display of dynamics systems, the I/UCRC at The University of Iowa has developed the Visualization of Dynamic Systems (VDS) program. It requires the simulation-updated position and orientation data specified by three translation values and Euler parameter vectors. The fidelity of the graphics animation is realistic enough to provide the operator with visual cueing comparable with TV cameras and video display screens. Furthermore, the animation can also be displayed in a split screen mode, or in two screens with moving view points to enhance depth and parallax perception.

8. Conclusion

Since flight simulators have proven cost effective for pilot training, their usage has been widely accepted throughout the airline industry. Now, a similar potential is on the horizon for mechanical systems. New recursive formulations for general purpose multibody dynamics simulation combined with high-speed parallel processing

computers are now capable of creating real-time man-in-the-control-loop simulators in a cost-effective manner. Such a simulator of the RRC robot arm is to be built in support of the FTS program. Its usage is not only for crew teleoperation training, but also for man-machine interface studies aimed at enhancing the ergonomic design of the telerobot and controller. The methodology is easily adapted to new or modified system design concept or control algorithm. It can also be used as a valuable tool for the study of human cognitive and behavioral science issues, as they apply to the telerobotic system.

REFERENCES

[1] Bae, D.S., Hwang, R.S., and Haug, E.J., "A Recursive Formulation for Real-Time Dynamic Simulation," Advances in Design Automation, ASME, DE-Vol. 14, September 1988, pp. 499-508.

[2] Hwang, R.S., Bae, D.S., and Haug, E.J., "Parallel Processing for Real-Time Dynamic System Simulation," Advances in Design Automation, ASME, DE-Vol. 14, September 1988, pp. 509-518.

[3] Haug, E.J. and McCullough, M.K., "A Variational-Vector Calculus Approach to Machine Dynamics," J. of Mechanisms, Transmissions, and Automation in Design, Vol. 108, 1986, pp. 25-30.

[4] Karlen, J.P., Thompson, J.M., and Farrell, J.D., 1987, "Design and Control of Modular, Kinematically-Redundant Manipulators," Second AIAA/NASA/USAF Symposium on Automation, Robotics and Advanced Computing for the National Space Program.

[5] Dubetz, M. and Haug, E.J., "Real-Time Dynamics Simulation: A Design Optimization Tool," Technical Report R-5, Center for Simulation and Design Optimization of Mechanical Systems, The University of Iowa, 1988.

[6] Dubetz, M. and Kuhl, J. "A Network Implementation of Real-Time Dynamic Simulation with Interactive Animated Graphics," Advances in Design Automation, ASME, DE-Vol. 14, September 1988, pp. 519-524.

PANEL ON GRAPHIC OVERLAYS IN TELEOPERATION

Graphic Overlays in High-Precision Teleoperation: Current and Future Work at JPL

Daniel B. Diner, Ph.D.
Steven C. Venema, B.A.
Automated Systems Section
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

ABSTRACT

In high-precision teleoperation, high-resolution visual depth information may be critical, thus requiring vision system capabilities quite different from lower precision teleoperation vision systems. Several possible approaches to providing this depth information are available. Multiple-camera television systems, 3-D television systems, and 3-D video graphics systems all have advantages and disadvantages.

Multiple camera TV systems provide depth information by providing several views of the workspace. In such systems, camera mobility is desirable. However, moving cameras can confuse the operator. Therefore, the operator must know at all times the location of each camera. Providing such information can be cumbersome and increase operator workload.

Converged stereo TV cameras configured for high-depth precision can yield significant depth distortions, thus making many high-precision tasks extremely difficult, even for trained operators.

Video graphic systems can provide depth information through a variety of techniques including monocular depth labeling by color, brightness, perspective, occlusion, etc., as well as traditional 3-D binocular image presentation. However, video graphics systems have a problem which TV systems do not have; i.e., when viewing unpredictable situations, graphics systems may not be able to provide critical information in a timely manner.

In space teleoperation additional problems arise, including signal transmission time delays. These can greatly reduce operator performance.

Recent advances in graphics open new possibilities for addressing these and other problems.

At JPL, we are currently developing a multi-camera system with normal and 3-D TV and video graphics capabilities. Trained and untrained operators will be tested for high-precision performance using two force-reflecting hand controllers and a voice recognition system to control two robot arms and up to 5 movable stereo or non-stereo TV cameras. Through extensive experimentation, we plan to evaluate a number of new techniques of integrating TV and video graphics displays to improve operator training and performance in teleoperation and supervised automation.

INTRODUCTION

Video graphics has recently advanced quite rapidly. Today, high-resolution, real-time graphic systems can be purchased off the shelf, thus establishing graphics as a candidate for real-time video image enhancement in high-precision teleoperation.

As the fields of robotics and teleoperation continue to develop, an increasing number of tasks which currently must be performed manually will be performed either remotely or under automation. Video graphics, a display technique for human observers, will most probably extend the capabilities of teleoperation more than robotics.

Graphics will be very useful to remote operators by providing information which would otherwise not be readily available, such as camera locations, repair manual diagrams, visual depth information, velocities of relevant objects on a video monitor, force-torque diagrams, etc.

In space, many of the tasks which are currently performed by EVA (extra-vehicular activity) will be performed in the future by IVA (intra-vehicular activity). This makes teleoperation and robotics extremely interesting to NASA.

Also, current EVA tasks which have traditionally been labeled as future robotic tasks may be accomplished sooner in the future under graphics-aided teleoperation. As time passes, the "division of labor" between robotics and teleoperation will be more clearly defined.

In this paper, we describe the future vision system of the Man-Machine Systems Research Lab at JPL. This lab is not to be confused with the Telerobot Demonstrator Testbed, described elsewhere in this conference.

BACKGROUND

When viewing a work space remotely, through a TV camera, the one imprecisely-displayed dimension is depth (i.e., distance from the TV camera.) This dimension is a critical requirement for good teleoperation.

Much work has been done on presenting the video depth information in 3-D stereo (1 - 10). High-precision, close-up, 3-D TV has been shown to have a depth-resolution/depth-distortion/image-alignment trade-off (7).

An alternative to 3-D TV is the use of a multiple-camera viewing system, where the depth information can be figured out by the operator by looking at the work space from several views simultaneously.

A third alternative is to use graphics information to provide depth information (10). Combinations of the above three depth display techniques are also feasible. Multiple 3-D views with graphics overlays promise to be very useful in teleoperation.

DISCUSSION

Current Work at JPL

Over the past 3 1/2 years, we have studied 3-D TV, both mathematically and experimentally. We have quantified the depth distortions, both for still and moving stereo camera rigs (7). We have found an optimal method of moving the stereo camera rig to minimize 3-D depth distortions caused by camera motions (8).

We have also demonstrated a stereo image presentation technique which yields aligned images, high depth resolution and low depth distortion, thus solving the trade-off problem (9,11). NASA has a patent on this technique.

Future Work at JPL

Although our stereo image presentation technique promises to enhance high-precision 3-D TV, multiple-camera viewing systems may still have an important role to play in the future of teleoperation, particularly with the addition of graphics. Single-camera stereo systems (11) provide the possibility of multiple stereo 3-D views.

We are building an experimental telerobotic work station with two robot arms surrounded by 5 movable TV cameras. The cameras will each be mounted on a computerized gantry frame, with one camera on each of the five sides of the gantry frame. That is, the front, the back, the left, the right and above. Each camera will have the ability to move in its plane (up-down and front-back for the two sides, up-down and left-right for the front and back, and left-right and foward-back for the top camera). In addition, each camera will be able to pan, tilt and change the power of the lens (zoom). Thus each camera can view the work space from any location and angle in its range of motion in its plane. Camera motions may be commanded by the operator, for example, using voice control, or may be automated, following the robot grippers as they move about the work space. We envision automating the system to tailor camera motions to the current task at hand.

Up to five monitors will be available for the five camera views. Two additional monitors may be available for system information, trouble shooting, etc. An image enhancement system will also be present which will include graphics capabilities, and perhaps image processing capabilities. The operator will be able to command (by voice control) which camera view will be displayed on each monitor. Initial configuration may be fixed, for example left camera on the left monitor, etc. This however is not required.

Our approach is both theoretical and experimental. The critical question, as always in this work, is operator performance. Experimentation alone can answer if operators perform better under one set of conditions than another. We intend to address a variety of topics in our research, including the following.

1. Camera Locations and Apparent Motion

When viewing a workspace with movable cameras, an operator can be greatly confused by not knowing at all times the locations, orientations, and motions of each camera. Apparent motion, when one believes that the world is moving when actually the camera is moving, is particularly confusing. When multiple cameras are available, the additional problem arises of knowing which camera view is presented on the monitor (or each monitor if there are multiple monitors). Graphics can help solve these problems by providing the necessary information.

We envision presenting a camera's video image with overlayed graphics information showing the location and orientation of the TV camera on the monitor. See Figure 1.

In Figure 1, the TV camera image shows the right robot holding a ball and the left robot holding nothing. In addition to the TV camera view is a top-view graphics image of the camera frame, showing the positions and orientations of the camera. In this configuration (top view) it is necessary to specify the height of the camera, perhaps with 3-D stereo depth, or some other form of depth labeling. The pan of the camera is obvious, and the tilt can be displayed graphically by lines and circles. For example, lines can mean 15 degrees elevation (front of camera above back) and pairs of circles can mean 15 degrees downward elevation. In Figure 1, the camera is tilted 45 degrees upward.

This graphics presentation can also be displayed on a separate monitor.

The advantage of this presentation is that although both robot grippers seem to appear at equal height, the fact that the camera is tilted upward tells us that, in fact, the left robot gripper is actually higher than the right gripper. Because we know that the camera is tilted 45 degrees

upward, we can judge better what motion will be necessary to hand the ball from the right robot gripper to the left robot gripper. If the TV image is stereo, then one can also judge the length of the motion.

Circles and lines need not be the best graphic illustration of tilt and, in fact, one of the variables we plan to research is how to best present the camera locations. "Best" is measured with respect to operator performance under a variety of tasks.

Another variable is the point of view of the graphics camera frame. In Figure 1, if the frame were presented from the side view, instead of the top view, both of the camera's translational degrees of freedom would be specified without depth labeling. Although this seems to be an obvious improvement, it may not be so. The top view unambiguously specifies which camera we are viewing through. In addition, with multiple monitors, operators may prove to perform better if all camera locations are specified from the same view.

Another alternative is to present all the cameras' locations on one graphics display of the camera frame. This would allow operators to use the graphics information and the voice controller to move a camera before viewing through it, thus saving valuable operator time. When using a system with several cameras, but only one monitor, moving a camera before viewing through it can be particularly valuable.

In a system where the lighting is variable, that is lights can be moved or turned on and off, the graphics can be used to specify the current state of the lighting system. The lighting can then be adjusted by voice control. In fact, any variable part of the system can be so specified, and adjusted.

Eventually, we plan to automate the system to control the cameras and graphics to provide the optimal view for each task during operation.

2. Image Jitter During Camera and Robot Motion

One may find it desirable for the camera to track the end-effector of the robot during robot motion. This raises the question of image jitter. Quite simply, if the camera does not move smoothly enough, or if the camera is not synchronized with the robot motion, the image of the robot will jitter on the monitor. Jittering images not only make precision operation difficult (one may want to tighten a bolt as the robot moves a unit across the workspace), but can increase operator discomfort.

We have designed our robot gantry so that the cameras can track the robot without jitter, provided only panning and tilting camera motions are used in the tracking. The maximum speed for jitter-free tracking is about 15 degrees/second. In our work configuration, that translates to robot motions of 15 to 70 cm/sec, depending on which camera is being used for tracking and the zoom setting of the lens. Thus, our system promises to provide excellent robot tracking capabilities.

3. Camera Motions and Coordinate Transformations

In a teleoperator work station, where movable cameras are viewing the work space, any panning, rolling or tilting of the cameras causes a mis-alignment between the coordinate system of the camera and the coordinate system of the operator viewing the monitor. For example, if the camera rotates 15 degrees to the left, the "straight ahead" direction on the monitor will actually be 15 degrees to the left. If one pushes a robot hand controller "forward", the robot will move foward, but will be seen on the monitor to move at an angle of 15 degrees to the right. This requires the operator to mentally transform coordinates continually, during operation, thus causing an increase in workload as well as an increase in the probability of operator error. If several movable cameras are presenting their images to several monitors, each may require a different coordinate transformation. The resulting increase in workload and

probability of operator error may well become unmanageable and dangerous.

When viewing a workspace with a movable camera, at least 7 coordinate systems exist: the Real World, the Work Space, the Robot Base, the Robot Joint, the Camera, the Control Station, and the Operator coordinate systems.

The problem then is to minimize operator workload produced by the transformations between these coordinate systems.

If the Robot-Camera Table is mounted on a moving vehicle, such as a planetary rover, then the Real World and the Work Space coordinate systems are different. If, however, the robot-camera table is not movable, then the Real World and the Work Space coordinate systems are equal. If the robot can move its base on the robot-camera table, then the Work Space and the Robot Base coordinate systems are different. If, however, the robot cannot move with respect to the robot-camera table, then the Work Space and the Robot Base coordinate systems are equal.

We use the term "Robot Base" coordinate system to distinguish from the Robot Joint coordinate system which customarily means the joint angles of the robot, and is different from the spatial (X,Y,Z,Pan,Tilt,Roll) coordinate system as defined from a fixed point on the robot, such as the robot base. The Robot Joint coordinate system is transformed to and from the Robot Base coordinate system by the software that controls the Robot, and is used by the robot's internal controller to move the robot joints correctly. Therefore we need not concern ourselves with the Robot Joint coordinate system here.

The Camera coordinate system is defined by what the camera sees. Thus, a camera panned to face southeast sees southeast as straight ahead. A camera rolled 180 degrees sees the earth as "up" and the sky as "down."

The Control Station coordinate system is defined with respect to the operator control station. Thus if the camera faces 15 degrees to the left in the Work Space coordinate system, then the direction straight ahead in the Work Space would be presented at 15 degrees to the right in the Control Station coordinate system.

The Operator coordinate system is defined with respect to the "subjective straight ahead" direction of the operator. A great deal of study has been conducted on this phenomena (12). For simplicity, let us assume that our operator defines this direction with respect to the operator control station, that is, the operator aligns himself or herself to face the control station directly. For now, we shall ignore the possibility that an operator may sit at an angle to the control station and not realize it.

At this point, let us consider a non-movable robot-camera table with a robot whose base is fixed to the table. Then the Real World, Work Space, and Robot Base coordinate systems are equal.

Our concerns then become the transformations between the Robot Base, the Camera, the Control Station, and the Operator coordinate systems. Let us see how they interact.

When a camera moves, say 15 degrees pan to the left, the Robot Base, the Control Station, and the Operator coordinate systems do not change. Only the Camera coordinate system changes; that is, straight ahead on the camera is now 15 degrees to the left for the Robot Base, the Control Station, and the Operator. Thus, motions directly away from the camera (directly into the monitor) are 15 degrees to the left w.r.t. all the other coordinate systems.

We believe that we have found a solution to the coordinate transformation problem, using graphics. JPL and NASA are currently considering patent rights on this method, and thus we cannot discuss it. If our idea is truly a solution, then its application will give the Robot Base, the Camera, the Control Station, and the Operator coordinate systems the same orientations. No transformations will need to be made by the operator, and no camera angles will need to be

remembered.

4. Orthogonal and Perspective Camera Views

We shall test operator performance with both orthogonal multi-camera views and perspective camera views. Let us discuss first the orthogonal-camera configuration.

Consider 3 cameras, one looking from above, one from one side, and one from the front. Consider 3 monitors placed with the top view above the front view, and the side view alongside the front view. This is the TV approximation to the classic orthogonal projection of mechanical drawings.

We say "the TV approximation" because it will not give a true orthogonal projection. In a TV image, two lines overlap if they point directly toward the camera, but in orthogonal projections, two lines overlap if they are perpendicular to the projection. See Figure 2. Thus, in fact, only the central line of view in the side camera is truly orthogonal to the front camera view. For the rest of the image, equal depth must be inferred. Two objects at equal depth from the front camera will have their front edges overlap exactly in the side camera's view only if the front edges of the two objects are viewed exactly at the middle of the side camera. The images of all other pairs of objects at equal depth will not overlap exactly. This is an important difference, and may prove to be the source of many operator errors when using orthogonal TV cameras. This point must not be overlooked, because it illustrates that orthogonal TV viewing may be misleading, particularly to people accustomed to orthogonal mechanical drawings, because they expect overlap to mean equal depth.

Let us now consider perspective viewing. This is the depth-display technique of the great Renaissance artists.

The left-brain/right-brain dichotomy between people suggests that people fall into analytic and artistic categories, particularly in terms of perception and motor performance. It also suggests that all of us have both artistic and analytic information processors in our heads. In any case, it is safe to say that we all have varying degrees of skill in judging depth both from orthogonal and perspective displays.

Unfortunately, orthogonal TV viewing has the problem discussed above. Thus, in multi-camera viewing, we may better perform using our perspective processor to judge depth. Surely, this needs to be tested experimentally, and carefully. We must first search for perspective views, and then test them against optimal orthogonal views.

5. Other Planned Graphics Overlay Experiments

We plan to test operator performance when aided by a variety of graphics overlays, including predictive displays and force-torque diagrams.

Predictive displays of robot positions are particularly useful when dealing with significant signal transmission time delays. When signals must travel long distances, for example through space, time delays between the time of an event and the time one views the event become significant. In a feedback loop, such as long-distance teleoperation, the time delay can greatly reduce performance.

Consider a teleoperated servicer (with a robot arm) on the moon, which is being controlled from earth. A time delay of about 4 seconds round-trip from the earth to the moon and back can be expected. Suppose at time $t = 0$, an operator moves the hand controller. At time $t = 2$ seconds, the servicer receives the signal and initiates the motion. At $t = 4$ seconds, the servicer is first seen to move on the operator's monitor.

With a predictive display, the expected final position of the robot arm is displayed as a graphics overlay on the monitor immediately after the hand controller is moved. This has been

described in detail elsewhere (13 - 16). For large time delays, the predictive display has been shown to improve operator performance (13). For small time delays, the extra information on the monitor from the predictive display may clutter the image and reduce operator performance. We shall test this question for a variety of tasks.

Force-torque displays graphically show the forces and torques sensed by the robot (17), at, say, the wrist. We shall test operator performance while varying the locations, size, and other presentation characteristics of the display. For example, we plan to overlay each robot's force-torque display on its forearm surface seen in the TV monitor. We shall also present the display on another monitor. Our goal, as in all our work, is to present the relevant information to the operator in a manner which increases operator performance.

CONCLUSION

Recent advances in graphics now make graphics a useful tool for enhancing video displays in teleoperation. At JPL, we are currently building a multi-camera viewing system with graphics capabilities. We plan to address certain problems in teleoperation that, once resolved, promise to enhance the capabilities of teleoperation. Our goal is to maximize the utility of teleoperation in space applications.

ACKNOWLEDGEMENTS

We thank Antal K. Bejczy, Derek H. Fender and Edward R. Snow for participating in many discussions during the process of this work. This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

REFERENCES

1. Shields NL, Kirkpatrick M, Malone TB and Huggins CT: *Design Parameters for a Stereoptic Television System based on Direct Depth Perception Cues.* Proceedings of the Human Factors Society 19th Annual Meeting (1975), Washington, D.C.

2. Grant C, Meirick R, Polhemus C, Spencer R, Swain D and Tewell R: *Conceptual Design Study for a Teleoperator Visual System Report.* Martin Marietta Corporation Report, NASA CR-124273 (1973), Denver, Colorado.

3. Upton HW and Strother DD: *Design and Flight Evaluation of a Helmet-mounted Display and Control System.* In Birt JA & Task HL (Eds.): "A symposium on visually coupled systems: Development and application", Brooks Air Force Base Report AMD-TR-73-1 (1973).

4. Zamarian DM: *Use of Stereopsis in Electronic Displays: Part II – Stereoscopic Threshold Performance as a Function of System Characteristics.* Douglas Aircraft Company Report MDC J7410 (1976).

5. Pepper, RL, Cole RE, and Spain, EH: *The Influence of Camera Separation and Head Movement on Perceptual Performance under Direct and TV-displayed Conditions.* Proceedings of the Society for Information Display (1983).

6. Bejczy, AK: *Performance Evaluation of Computer-aided Manipulator Control.* IEEE International Conference on Systems, Man and Cybernetics (1976).

7. Diner DB and von Sydow M: *Stereo Depth Distortions in Teleoperation.* JPL Publication 87-1 (1987), Jet Propulsion Laboratory - NASA, USA. National Technical Information Service # 87N18985; Proceedings of the Twenty-Second Annual Conference on Manual Control (1986); AFWAL-TR-86-3093, Wright-Patterson AFB Aeronautical Labs, Ohio, USA.

8. Diner DB and von Sydow M: *Dynamic Stereo Vision Depth Distortions in Teleoperation.* Proceedings of the International Symposium on Teleoperation and Control (1988), Springer - Verlag, New York. ISBN 0-387-50054-5.

9. Diner DB and von Sydow M: *Static Stereo Vision Depth Distortions in Teleoperation,* "Ergonomics of Hybrid Automated Systems I" (1988), Elsevier Science Publishers B. V., Amsterdam.

10. Fisher SS, McGreevy, Humphries H, Robinett W: *Virtual Environment Display System* NASA Ames Research Center, IEEE Computer Graphics 21:1 (1987).

11. Diner DB and Fender DH: *Handbook of Human Engineering in Stereoscopic Viewing Devices* (In preparation). JPL External Publication, Jet Propulsion Laboratory, Caltech, Pasadena, California, USA. Expected Publication 1989.

12. Howard IP: *Perception of Posture, Self Motion and Visual Vertical Handbook of Perception and Human Performance I.* John Wiley and Sons, Inc., New York (1986).

13. Hashimoto T, Sheridan TB and Noyes MV: *Effects of Predictive Information in Teleoperation with Time Delay.* Japan Journal of Ergonomics 22,2 (1986).

14. Noyes MV and Sheridan TB: *A Novel Predictor for Telemanipulation Through a Time Delay.* Proceedings of Twentieth Annual Conference on Manual Control (1984), NASA Ames Research Center, Moffett Field, California.

15. Buzan FT: *Control of a Teleoperator in the Presence of Large Time Delays.* MIT Man-Machine Systems Laboratory, JPL Contract no. 956892, Mid-Term Reports, June (1986) and May (1987).

16. Venema SC and Bejczy AK: *The Phantom Robot: Predictive Displays for Teleoperation with Time Delay.* IEEE International Conference on Robotics and Automation (1989). Submitted.

17. Bejczy AK and Dotson RS: *A Force-torque Sensing and Display System for Large Robot Arms.* Proceedings of IEEE Southeastcon82, Destin, Florida (1982).

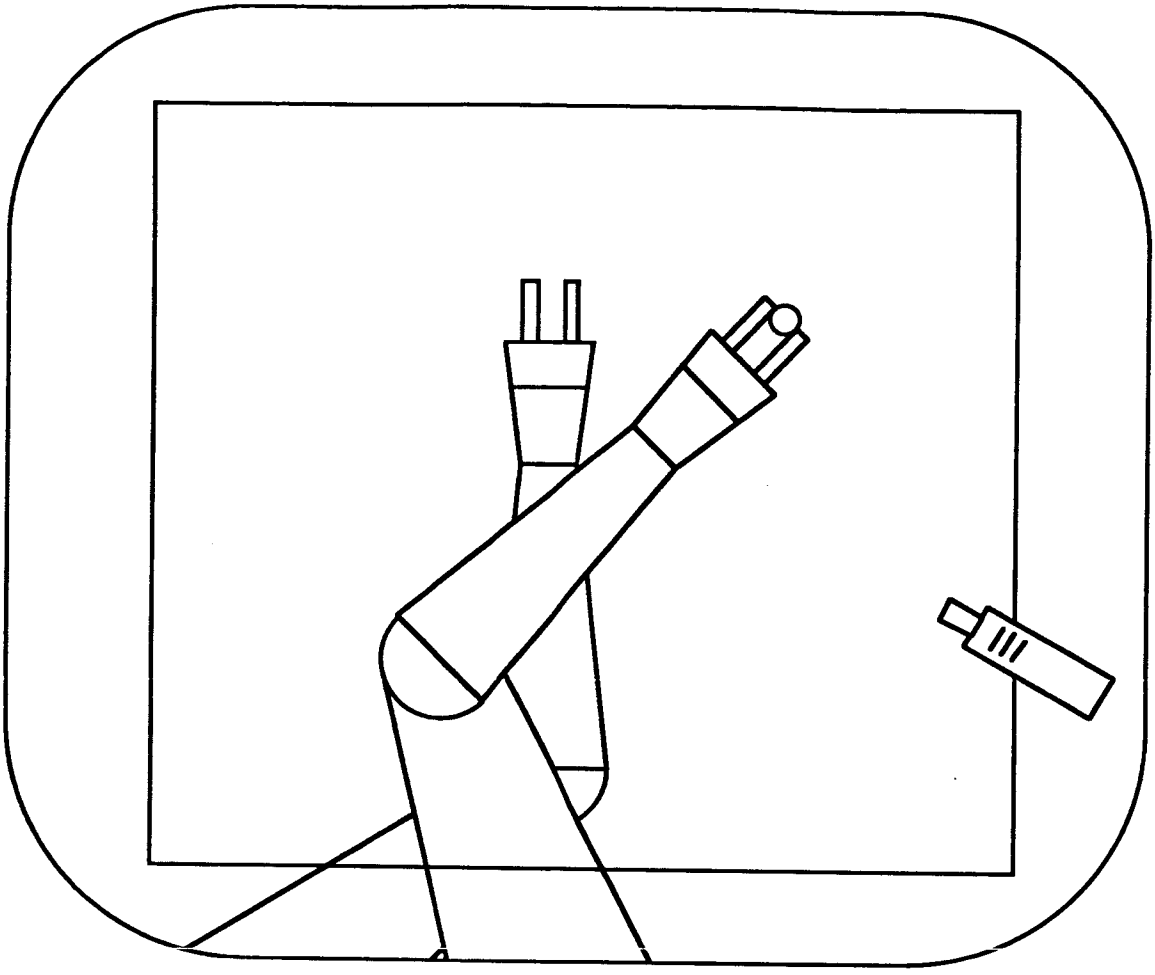


Figure 1: TV camera image of two robot arms with graphic overlay of top-down view of camera frame and camera location.

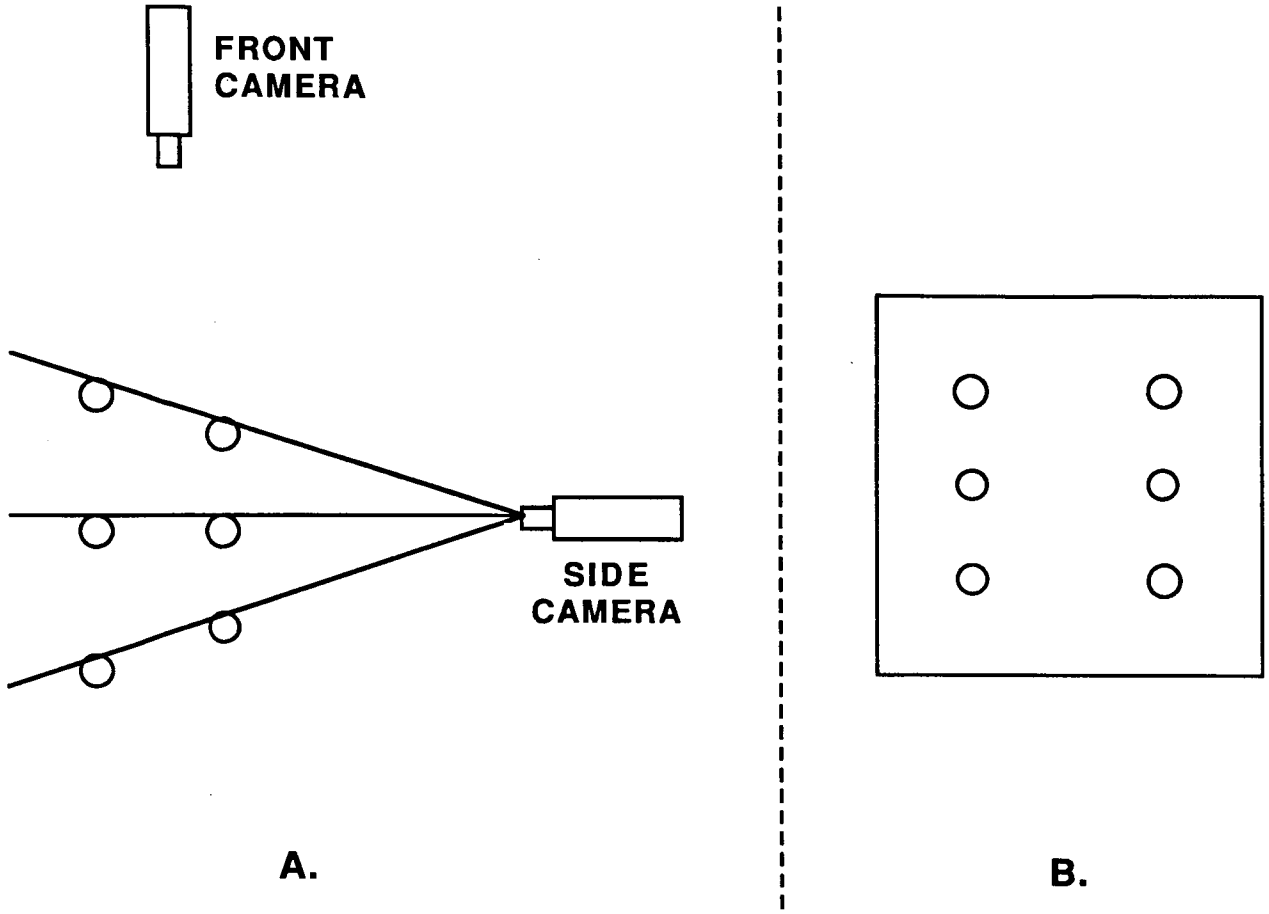


Figure 2: Top view of the locations of 3 pairs of objects which:
(a) overlap in a side TV-camera view, and
(b) overlap in a side view in standard orthogonal mechanical drawings.

Head-mounted spatial instruments II: [1] synthetic reality or impossible dream

Stephen R. Ellis
NASA Ames Research Center
Moffett Field, California USA
and
University of California
Berkeley, California

Arthur Grunwald
TECHNION
Haifa, ISRAEL

Abstract

A spatial instrument is defined as a spatial display which has been either geometrically or symbolically enhanced to enable a user to accomplish a particular task. Research we have conducted over the past several years on 3D spatial instruments has shown that perspective displays, even when viewed from the correct viewpoint, are subject to systematic viewer biases. These biases interfere with correct spatial judgements of the presented pictorial information. The design of spatial instruments may not only require the introduction of compensatory distortions to remove the naturally occurring biases but also may significantly benefit from the introduction of artificial distortions which enhance performance. These image manipulations, however, can cause a loss of visual-vestibular coordination and induce motion sickness. Consequently, the design of head-mounted spatial instruments will require an understanding of the tolerable limits of visual-vestibular discord.

Introduction

The introduction of relatively low cost, interactive, high performance 3D computer graphics work-stations such as the Personal IRIS or the Megatek 928, and the certain prospect for further miniaturization and cost reduction, has provided aerospace designers with powerful research tools for creating new media for interactive, information displays.

This flexibility raises many practical design challenges and interesting theoretical questions, but since many of these new information displays may be helmet or head mounted, particularly prominent questions concern guaranteeing the perceptual stability of the display's image. Indeed, it is argued in this paper that selecting a head-mounted format limits design freedom in the definition of the displays in ways that do not constrain conventional panel-mounted formats.

Analysis

An understanding of the relevant design questions is best provided by an analysis of the linear transformations that the spatial information must undergo before presentation to the user. In general, the information is first defined as sets of vectors, polygons, or polyhedra positioned in an inertial reference frame some times called the "real world" coordinate systems. (Foley and Van Dam, 1982).

Prior to presentation to the viewer, this information must be transformed by scaling, rotation, translation, and projection to position it in an "eye coordinate system" determined by the position and direction of a viewing vector. This transformation processes is commonly represented as a series of matrix operations and is referred to as the "viewing transformation", but as shown in Figure 1, it may be broken into several separable parts each of which allows a unique opportunities for the introduction of informative distortions.

Subsequent use of this spatial information by the viewer requires that he internally perform further coordinate transforms to bring it into a useful frame of reference. For example, if the subject is required to make an egocentric direction judgment based on information on a 3D map, he must further

[1]Earlier versions of this manuscript have been reported at the 1987 AGARD Meeting of the Aerospace Medicine Panel in Brussels, Belgium, September 28 - October 2, 1987 and at the 1988 California Mapping Conference, San Jose, California.

transform the information into a body or even a hand centered coordinate system by a process similar to the viewing transformation. These are the transformations typically required in telerobotics.

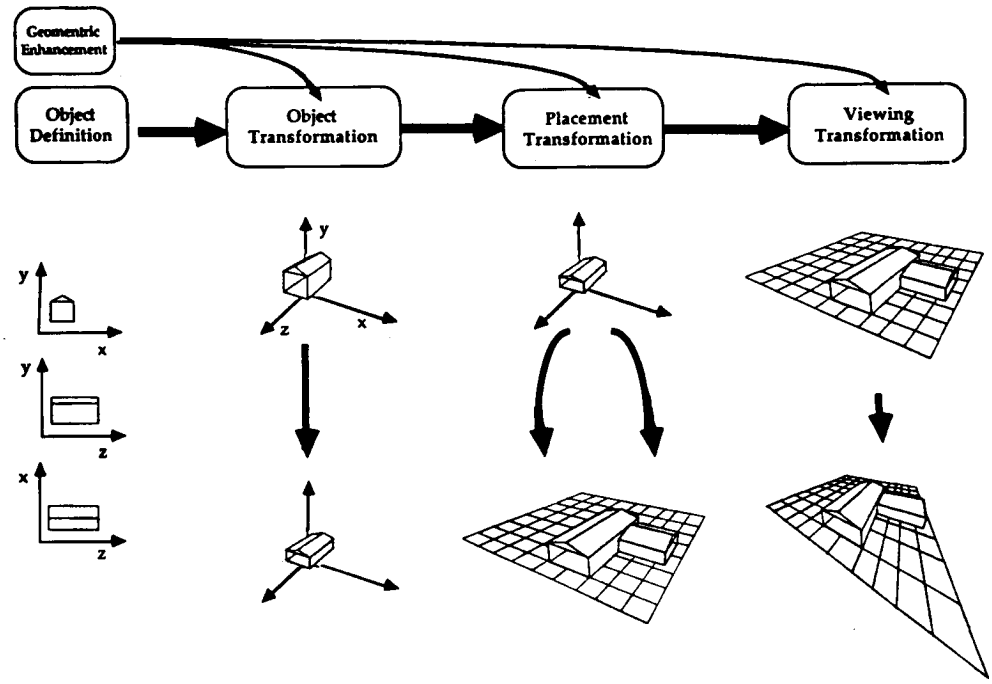


Figure 1. The process of representing a graphic object in the virtual space allows a number of different opportunities to introduce informative geometric distortions or enhancements. These may either be a modification of the transforming matrix during the process of image definition or they may be modifications of an element of a model. Often the matrix element or shape of the model part is controlled externally by a variable slewed to a mouse or other input device. These interventions may take place 1) in an object relative coordinate system used to define the object's shape or 2) in an affine or even curvilinear object transformation, or 3) during the placement transformation that positions the transformed object in world coordinates, or 4) in the viewing transformation. The perceptual consequences of informative distortions are different depending upon where they are introduced. For example, object transformations will not impair perceptual stability in a head-mounted display whereas manipulations of the viewing transformation will.

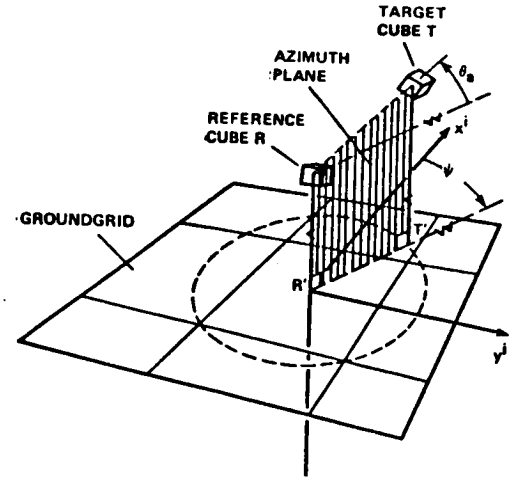


Figure 2. The relative direction of one cube with respect to another and a reference direction x is given by the difference in the judged egocentric azimuth rotation of two objects: the ground grid which provides the reference and the azimuth plane defined by perpendiculars dropped from the cubes to the grid. In order for a viewer to perceive the exocentric direction Ψ of the target cube he must recover the viewing parameters used to make the picture.

In order to understand how the spatial information presented in pictures may be used, it is helpful to distinguish between images which may be described as spatial displays and those that were designed to be spatial instruments. One may think of a spatial display as any systematic mapping of one space onto another. A picture or a photograph is a spatial display.

A spatial instrument, in contrast, is a spatial display that has been enhanced either by geometric or symbolic techniques to insure that the communicative intent of instrument is realized. A simple example of a spatial instrument is an analogue clock. In a clock the angular positions of the arms are made proportional to time, and the viewer's angle estimation task is assisted by radial tic marks designating the hours and minutes. A second aspect of the definition of a spatial instrument, which the clock example also illustrates, is that the communicated variable, time, is made proportional to a spatial property of the display, such as an angle, area, or length and is not simply encoded as a character string.

The spatial instruments that we wish to focus attention on are generally interactive. That is to say that the communicated information flows both to and fro between the viewer and the instrument. Some of this bidirectional flow exists for practically all spatial instruments since movement of the viewer's viewpoint can have a major impact on the appearance of the display. However, the displays we wish to consider are those incorporating at least one controlled element, such as a cursor, which is used to extract information from and input information to the instrument.

Maps also meet the definition of a spatial instrument. The map projection may be chosen depending upon the spatial property of importance. Choice of this projection illustrates objective geometric enhancement. Overlaying of a graticule of latitude and longitude lines indicating the map metric is an example of symbolic enhancement. When fitted with these enhancements, the map can become a nomographic calculating instrument for navigation or spatial orientation.

In selecting a map projection for a small scale map, such as a world map, a cartographer may select a projection from three families of perspective projections in which the solid angle formed at the point of contact between the globe and the projection surface varies 1) from 2π steradians, the zenithal case, 2) from 2π up to 0 steradians, the conical case, and 3) 0 steradians, the cylindrical case.

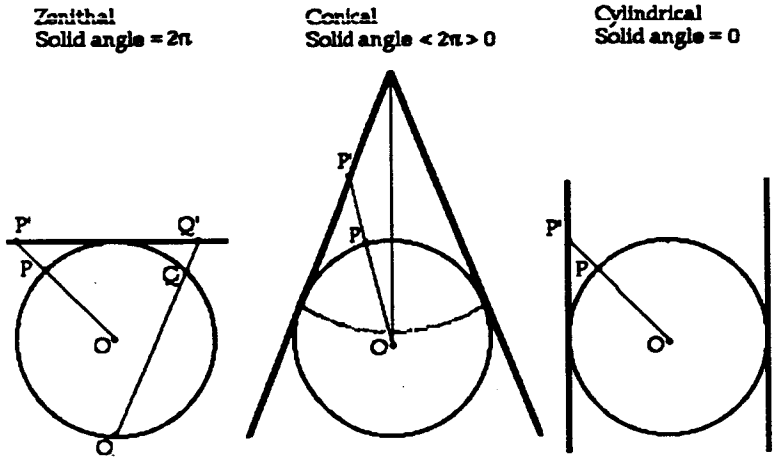


Figure 3. Geometric categories of map projections.

This selection can be guided by the ultimate use of the map. If, for example, a map is to be used to find minimum distance routes between distant points, a special case of the first type, the gnomonic projection can be used since it has the useful property of projecting all great circles as straight lines. (See Figure 4) One corresponding cost that must be incurred for this useful property is that the nonlinear scale distortion along meridians and parallels is unequal. Scale exaggeration for a piece of meridian in

the vicinity of latitude θ is $cosec^2\theta$ and exaggeration of latitude scale is $cosec\theta$. The map is, thus, not orthomorphic, i.e. shape preserving (Cotter, 1966). Nonlinear scale distortions of this sort are, however, well understood and can be objectively controlled by selection of the point of contact of the projection and the extent of area represented on the map.

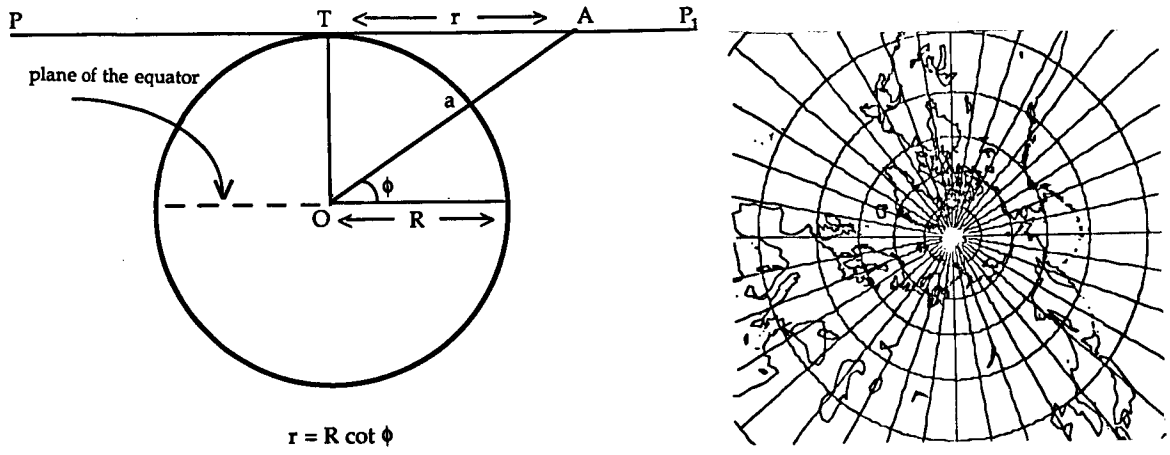


Figure 4. Geometry of radial scale exaggeration in a gnomonic projection with an example of a gnomonic projection centered on the north pole.

The distortions present on the gnomonic projection are a geometric consequence of the selected perspective parameters which describe the projection of the of the globe onto a tangent projection surface. They are not explicitly introduced but rather are a side-effect of the desired property that great circles map as straight lines. Distortion, however, can also be introduced directly into a projection to achieve a desired end as in popular conventional projections such as the mercator chart which is designed to map compass courses as straight lines. In this projection, a rectangular projection with the standard parallel set to the equator is intentionally distorted along the meridians to compensate for the fact that the scale along each parallel of the rectangular projection represents a smaller and smaller small circle as higher latitudes are mapped. Since the circumference, r, of a circle of latitude θ on a globe of radius R is: $r = Rcos(\theta)$. each small element of the meridian must be stretched by $1/cos(\theta)$ to compensate and straighten out the plots of oblique courses. The resulting scale exaggeration near the poles is so great to make the map unusable, but the technique works well for middle latitudes.

The geometry of gnomonic or mercator projections is well understood and adapted to provide geometric properties on the respective maps that are objectively useful. The straight line plotting of either great circles or rhumb lines facilitated use of the maps for navigation since desired courses could be found with a straight edge. Today in the time of computer-graphics based dynamic maps this advantage is not nearly as important. As is clearly evident from the software-based tools for interacting with spatial data bases as used for computer aided design, the old fashioned ruler and pencil have been generalized into multidimensional probes and almost magical cursors that can quickly extract highly dimensional spatial information from a data base which earlier generations of draftsman could barely imagine (Silicon Graphics, 1988; Dickinson, 1989ab). Consequently, the desired objective properties of images of spatial data are now not the same and in fact may be less important than the subjective appearance of the space depicted in the image. It is this subjective appearance that most directly influences the users interaction with the spatial data through his control of a cursor. Accordingly, the subjective appearance of the image now can become an important design feature.

Spatial Instruments

Contemporary spatial instruments are found throughout the modern aircraft cockpit, the most notable probably being the attitude direction indicator or ADI which displays a variety of signals re-

lated to the aircraft's attitude and orientation with respect to earth based navigation aids. More recent versions of these standard cockpit instruments have been realized with CRT, cathode ray tubes, based instruments which have generally been modeled after their electromechanical predecessors (Boeing, 1983).

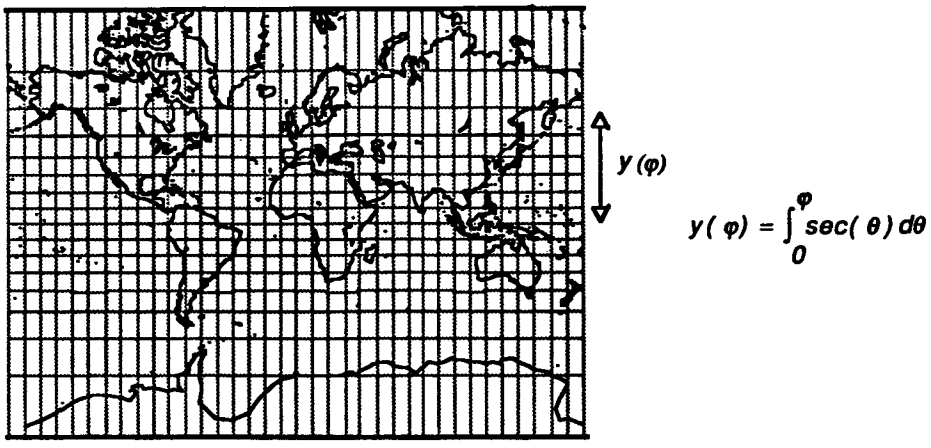


Figure 5. Compensatory distortion of a rectangular projection is illustrated by the change in scale along the meridians of the mercator projection.

The computer graphics and CRT display media, however, allow the conception of totally novel display formats for demanding new aerospace applications. Grunwald and Ellis (Grunwald and Ellis, 1988) have described, for instance, a more pictorial spatial instrument to assist informal, complex, orbital navigation, proximity operations, and rendezvous in the vicinity of the space station (see Figure 6). The definition of this instrument entailed a number of specific graphical enhancements which may be classified as either geometric, symbolic, or both. For example, a geometric enhancement was introduced by providing a display mode in which the axis along which spacecraft typically follow reentrant looped paths is transformed into a time axis which does not exhibit these loops. This transformation may assist obstacle avoidance and out of plane maneuvering during small orbital changes. The use of a time axis may also be a technique to avoid visual illusions associated with perspective projections of the trochoidal paths that describe the relative motion paths of one spacecraft with respect to each other.

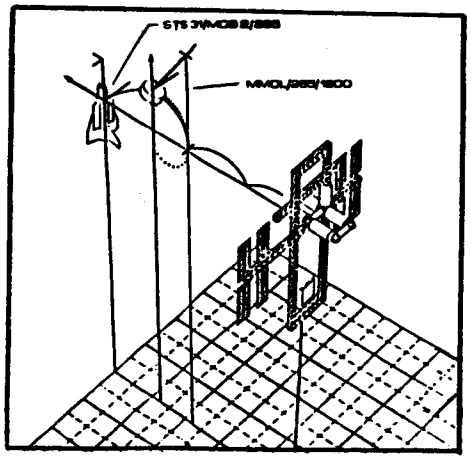


Figure 6. Sample proximity operations display. The solid curves lines show a planned orbital rendezvous between an orbital maneuvering vehicle (OMV) and the space station. The dotted line is a predicted flight path for the OMV. The projecting vectors show body axes of the craft.

Geometric Enhancement

In general, there are various kinds of geometric enhancements that may be introduced into spatial displays, but their common feature is a transformation of the metrics of either the displayed space or of the objects it contains. A more familiar example is found in relief topographic maps for which it is useful to exaggerate the vertical scale. This technique has also been used for experimental traffic displays for commercial aircraft. (Ellis, McGreevy, & Hitchcock, 1987)

Another type of geometric enhancement important for displays of objects in 3D space involves the choice of the position and orientation of the eye coordinate system used to calculate the projection. Azimuth, elevation and roll of the system may be selected to project objects of interest with a useful aspect. This selection is particularly important for displays without stereoscopic cues, but all types of displays can benefit from an appropriate selection of these parameters. (Ellis, Kim, Tyler, McGreevy and Stark, 1985; Kim, Ellis, Tyler, Hannaford, and Stark, 1987).

Because of its dramatic effect on the image, selection of the field of view angle is particularly interesting. Only changing the field of view angle simply magnifies the image producing image which corresponds to an optic array geometrically similar to that optic array that a viewer would experience from the modeled eye point. Selecting a very wide field of view angle results in a minimized image, but also can introduce marginal distortions if a planar projection surface is used to produce the image. An additional source of distortion can arise if the display is viewed from a point other than the modeled eye point in the eye coordinate system. The effects of these latter distortions may, however, be modulated by the viewer's awareness of the picture plane (Pirenne, 1970; Ellis, Smith, McGreevy, 1987).

Significant design features can be achieved by joint variation of the field of view angle as objects in the display (McGreevy and Ellis, 1986; Ellis, et al., 1987; Adams, 1975). Though this combined manipulation may introduce marginal distortions, it allows control over the projected sizes of objects in the image and, for example, allows definition of a projection that will always include a designated volume of the object space. This is a useful property of a situation awareness display which is not preserved in a display by changes in the field of view alone.

The introduction of deliberate spatial distortion into a spatial instrument can be a useful way to improve the communication of spatial information to a viewer since the distortion can be used to correct underlying natural biases in spatial judgements. For example, exocentric direction judgements (Howard, 1982) made of extended objects in perspective displays, can for some response measures exhibit a "telephoto bias". That is to say that the subjects behave as if they were looking at the display through a telephoto lens. This bias can be corrected by introduction of a compensating wide-angle distortion. (McGreevy and Ellis, 1986; Grunwald and Ellis, 1987)

Unnatural scaling by placement transformations can also be used to control an objects prominence, to insure, for example, that they never become vanishingly small. (see Figure 7). Scaling with an object transformation is also particularly effective at achieving nonlinear exaggerations but such unnatural object scaling can, however, increase display clutter: objects may interpenetrate. But independent scaling of the separate axes of the object generally provides the designer with techniques to reduce this interpenetration.

Symbolic Enhancement

Symbolic enhancements generally consist of objects, scales, or metrics that are introduced into a display to assist pick-up of the communicated information. The usefulness of such symbolic aids can be seen, for example, in displays to present air traffic situation information which focus attention on the relevant "variables" of a traffic encounter, such as relative altitude, as opposed to less useful "properties" of the aircraft state such as absolute altitude (Falzon, 1982).

One way to present an aircraft's altitude relative to a pilot's own ship on a perspective display is to draw a grid at a fixed altitude below the "ownship" symbol and drop reference lines from all air-

craft symbols onto the grid. If the "ownship" altitude is marked on these reference lines, then the distance from the other aircraft symbol to the mark is proportional to the relative altitude. If the aircraft are given predictor vectors that show future position, similar reference lines can be dropped from the ends of the predictor lines.

The reference lines not only serve to clarify the target's ambiguous aspect but also can improve perception of the target's heading. This effect has been shown in a recent experiment examining the effects of reference lines on egocentric perception of azimuth of extended objects in perspective images created by a microcomputer graphics system. This experiment provides a specific example of how psychophysical evaluation of display formats can be used to assess their information display effectiveness.

In this experiment 10 subjects viewed static perspective projections of aircraft-like symbols elevated at three different levels above a ground reference grid: a low level below the view vector and almost on the grid, a middle level co-linear with the viewing vector, and a high level above the view vector by the same amount as the low level was below it (see Figure). The aircraft symbols have straight predictor vectors projecting forward showing future position above the reference grid. In one condition reference lines were dropped only from the current aircraft position. In the second condition reference lines were dropped also from the ends of lines projecting from each aircraft. These lines could represent predictors of future position.

The subjects viewed the entire configuration of aircraft symbol and grid from a fixed eye position 28 cm from the projection surface. This position was at the geometricaly correct center of projection for a viewing vector set to 0 degrees azimuth and -22.5 deg elevation. Nine different azimuth rotations of the image were presented: 0 to 180 in 22.5 degree increments. The subject's task was to adjust the egocentric direction of a horizontal dial to indicate the azimuth rotation of the aircraft. Azimuth rotation was crossed with number of reference lines in a factorial repeated measures experiment.

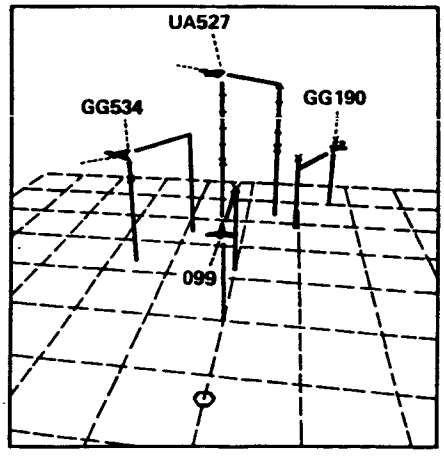


Figure 7. Sample cockpit display of air traffic. Own ship is at the center of the display. 1 minute predictors project out of all aircraft symbols. Reference line are dropped perpendicular to the reference grid.

The first result of the experiment was that subjects made a substantial errors in their estimation of the azimuth rotation of the aircraft; they generally saw it rotated more towards the picture plane than it in fact was.($F= 23.4$, $df = 8,72$; $p. < .001$) This corresponded to clockwise errors for actual clockwise rotations up to 90 degrees. The errors reverse for rotations greater than 90 degrees.

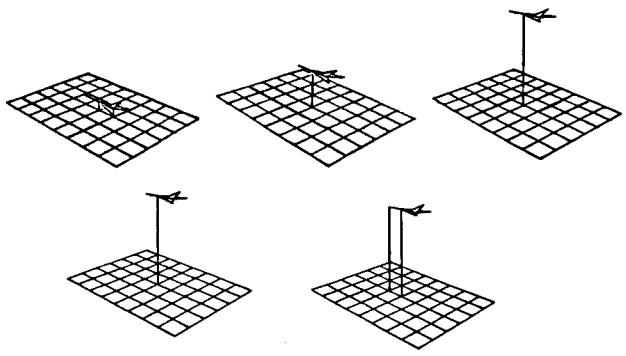


Figure 8. Five views of sample stimuli used which illustrate the three heights of the aircraft symbol above the grid and the two reference line conditions. Viewing elevation = -22.5 deg, azimuth = 45 degrees. As the aircraft rises away from the grid it develops an illusory yaw toward the picture plane.

The second result is that the the error towards the frontal plane for the symbols with one reference line increased as the height of the symbol increased above the grid ($F = 4.1$, $df = 2,18$, $p < .34$). Most significantly, however, as shown in Figure 9, introduction of the second reference line totally eliminated the effect of height, reducing the azimuth error in some cases about 50% ($F = 2.402$, $df = 16,144$, $p < .003$). A more detailed geometric and perceptual analysis of this result is beyond the scope of this paper; however, these experimental results show in a concrete way how appropriately chosen symbolic enhancements can provide not only qualitative but quantitative improvement in pictorial communication.

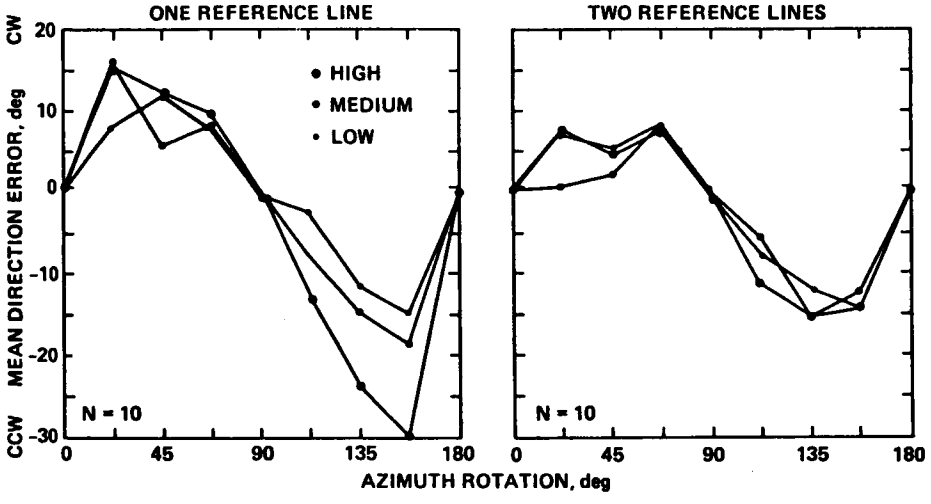


Figure 9. Mean clockwise and counterclockwise egocentric direction judgement for clockwise azimuth rotation.

Combined Geometric and Symbolic Enhancements

Some enhancements combine both symbolic and geometric elements. One good example is provided by techniques connecting the photometric properties of objects or regions in the display with other geometric properties of the objects or regions themselves. Russell and Miles (1987), for example,

have associated the optical density of points in space with the norm of the gradient of the concentration of a dissolved component and produced striking visualization of three-dimensional distributions of the compound. Similar techniques have been applied to solid models derived from sequences of CAT scans and allowed a kind of "electronic dissection" of medical images by control of the transparency of the different tissue types contained in the X-ray images (Meagher, 1987). Though this technique can provide absolutely remarkable images; one could for example "see the wind" by making optical density proportional to velocity; one of the challenges of its use is the introduction of metrical aids to allow the viewer to pickup quantitative information from the photometric transformation.

Discussion

The different types of enhancement are important in particular for head-mounted displays because they interact differently with the image and viewer. The global geometric enhancements are particularly important for head-mounted displays since they interfere with visual-vestibular coordination and can result in motion sickness.

Computer generated, helmet-mounted images were probably first produced by Ivan Sutherland in 1970 (Sutherland, 1970) and have more recently been produced somewhat more elaborately at several other laboratories. (Furness, 1986; Fisher, McGreevy, Humphries, Robinett, 1986). When Sutherland developed his display, the required hardware and software investment was substantial and available only to well funded laboratories. In contrast today, the display technology has become so inexpensive that a system adequate for creditable research can be assembled within a budget of a few thousand dollars.

Presentation of the computer generated image display on a head mounted display strongly encourages the viewer to interpret the projection as a virtual space which is expected to interact with his movements as if it were a real space. This kind of interpretation also occurs, but to a lesser extent, with ordinary pictures presented in the normal panel mounted format. The interpretation of a virtual space can give rise to pictorial illusions of depicted orientation (Goldstein, 1987; Ellis, Smith, and McGreevy, 1987), but these effects are far weaker with panel mounted displays than with those that are helmet-mounted.

One reason for the difference is that the helmet displays often include collimating optics, (Weintraub et al., 1985) producing true virtual images and interfering with viewers ability to locate the surface of the picture (Nagata, 1986). Furthermore, the helmet displays generally present wider fields than the panel mounted displays. These viewing conditions, which trigger the normal binocular reflexes associated with vergence accommodation, coupled with the vestibular effects of head movement result in a viewing situation that requires careful calibration to insure perceptual stability. If stereoscopic presentation or head driven motion parallax are used, this requirement is assured.

The difficulty with this format is that the global geometric enhancements destroy the required calibration. This difficulty is true by definition for the enhancements, such as differential scaling of the display axes, that operate on the viewing transformation itself, but it is also true, though to a lesser extent, of enhancements such as differential object scaling because familiar size can be the overriding cue to apparent distance (Ittelson, 1951). This effect may have operational significance and explain errors pilots make when using virtual image displays (Roscoe, 1984; 1987).

The loss of visual stability due to improper correlation between visual and vestibular movement arises from both voluntary and involuntary head movement. Large voluntary head movements can produce the most obvious loss of stability if the gains and phase lags between the image movement and vestibular ocular reflex (VOR) do not match. Fortunately, the VOR is adaptable and can adjust its gain and phase response (Bertoz and Melville-Jones 1985), though time lags resembling transport delays may preclude this adaptation. Small involuntary head movements cause relative movement between the head and the viewing axis of the eye which is inertially stabilized by the VOR. In this situation the head-mounted display screen moves and blurs the image. Thus the normal operation of the VOR is actually counterproductive. Measurement of the actual head movement can provide a signal to allow

compensatory, inertial stabilization of the display by displacement on the screen by adaptive filters which can model the VOR (Wells and Griffin, 1984; Velger *et al.*, 1988).

Besides loss of visual stability, geometric enhancements can interfere with visuo-motor coordination. This interference is particularly evident if the display includes a hand-controlled cursor. Under these circumstances an improperly calibrated or and intentionally distorted display resembles the view through a prism and lens system that introduces an optical distortion into the lines of sight. As known at least from the time of Helmholtz (1856), the visuo-motor system can completely adapt to the kind of conformal transformation such system can produce. Short time delays, on the order of 100 msec., can, however, substantially degrade or block this adaptation. (Held, Efstathiou, and Greene, 1966).

Allowable Enhancements for Helmet Mounted Instruments

In view of the many intrinsic problems with purely geometrical enhancement, the safest enhancements for helmet mounted instruments seem to be symbolic, the kind of added information overlays that have been used on aircraft head-up-displays for years.

These displays typically transpose much of the information already available in aircraft cockpits into a more integrated form and present it on a large combining plate, or beam splitter, so the information is available "head up" and can be seen when the pilot looks out the window (Weintaub, Haines and Randle, 1985). In addition to the usual moving tape, cursors, or numerical readouts, these displays often have a small graphics image projected to correspond in shape, size and position to an out-the-window object such as a runway. Maintaining good calibration for such an overlap between a display-generated graphics object and the projection of a real external object represents a significant challenge in a wearable helmet not using skull screws to maintain its position on the users head. Indeed, helmet mounted displays of this sort have been suggested as a useful nausea-inducing apparatus to attempt to habituate astronauts to the sensory discordance of weightlessness before they begin space travel. (Parker, Renschke, Arrott, Homick, and Lichtenberg, 1986).

Never the less, symbolic use of three-dimensions also seems to be an allowable enhancement. For example, one can imagine three-dimensional icons representing records in a hierarchical data base for which the third dimension could represent depth of nesting. Another interesting possibility for symbolic aid could be transient 3D "yardsticks" used in combination with a 3D cursor to designate pairs of objects to be compared. Once two objects are selected, a line symbolically designating their separation could be temporarily generated to display a binary relation between them.

Among the geometric enhancement, those least likely to cause visual stability problems are those that act on the real world coordinates of the displayed objects themselves: the object scaling transformations. Provided that the transformed objects do not markedly violate the viewers implicit assumptions about size and shape, these transformations act early enough so that their effect may interpreted simply as changing the shape and size of the objects. They would unfortunately interfere with manual manipulation of the objects, but as long as this is carried out symbolically with a cursor and not with a simulated "hand" with many degrees of control which must be adapted to the conditions of the display space, these size and shape transformation should not be too aversive.

Finally, the photometric transformation illustrated by Russell and Miles (1987) is unlikely to have untoward consequences for head mounted instruments and may prove useful if combined with metrical aids allowing them to present more quantitative information.

In the final analysis the basic limits in the definition of helmet mounted instruments may not be classically technological, but intellectual. The technological limits faced in the design of these tools will be foreseeably over come by time and effort of others involved with the seemingly inevitable progress of optical and electronic fabrication. The intellectual limits will be overcome only by designers imagination and understanding of human spatial perception.

References

Adams, Ansel. (1975) *Camera and lens.* Morgan and Morgan, New York

Bertoz, A. and Melville-Jones, G. (1985) *Adaptive mechanisms in gaze control: facts and theories.* Elsevier, New York.

Boeing (March, 1983) 757/757 Flight deck design development and philosophy, D6T11260-358. Seattle.

Cotter, Charles H. (1966) *The astronomical and mathematical foundations of geography* . New York, Elsevier

Dickinson, Robert (1989a) A unified approach to the design of visualization software for the analysis of field problems. *SPIE/SPSE Symposium on Electronic Imagining,* Conference 1083 vol 1083.

Dickinson, Robert (1989b) Interactive 4D visualization of fields with applications including meteorology, FEM, CFD, analyses and molecular fields. (Submitted for publication SIGGRAPH89.)

Ellis, S.R., Kim, Won Soo, Tyler, Mitchell, McGreevy, M. W. , Stark, L. (Nov., 1985). Visual enhancements for perspective displays: perspective parameters. *Proceedings of the 1985 International Conference on Systems Man and Cybernetics.* IEEE Catalog # 85CH2253-3, 815-818.

Ellis, Stephen R., Kin, Won-Soo, Tyler, Mitchell, Stark, Lawrence. (1985). *Proceedings of the 1985 International Conference on Systems, Man, and Cybernetics* (pp.815-818). New York: IEEE.

Ellis, Stephen R., Smith, Stephen, McGreevy, Michael W. (1987) Distortions of perceived visual directions out of pictures. *Perception and Psychophysics, 42,* 535-544.

Ellis, Stephen R., Grunwald, Arthur. (1987) A new visual illusion of projected three-dimensional space. *NASA TM 100006,* NASA Ames Research Center., Moffett Field, CA,

Falzon, P. (1982, June). Display structures: compatibility with the operators mental representations an reasoning processes. *Proceedings of the 2nd European Annual Conference on Human Decision Making and Manual Control.,* 297-305). Wachtberg-Werthoven, Federal Republic of Germany: Forschungsinstitut fuer Anthropotechnik.

Fisher, S.S., McGreevy, M.W., Humphries, J., Robinett, W. (1986) Virtual environment display system. ACM 1986 Workshop on 3D Interactive Graphics, Chapel Hill, North Carolina. October 23-24, 1986

Foley, J.D. and Van Dam (1982) *Fundmentals of interactive computer graphics.* Addision-Wesley, Reading, Mass.

Furness, Thomas F. (1986) The super cockpit and its human factors challenges. *Proceeding of the Human Factors Society, 30th Annual Meeting,* 48-52.

Held, Richard, Efstathiou, Aglaia, and Greene, Martha. (1966) Adaptation to displaced and delayed visual feedback from the hand. *Journal of Experimental Psychology, 72,* 887-891.

Ittelson, W. H. (1951) Size as a cue to distance: static localization. *American Journal of Psychology, 64,* 54-67.

Glickstein, Mitchel (September, 1988) The discovery of the visual cortex. *Scientific American.*

Goldstein, E. Bruce (1987) Spatial layout, orientation relative to the observer, and perceived projection in pictures viewed at an angle. *Journal of Experimental Psychology, 13,* 256-266.

Grunwald, Arthur, & Ellis, Stephen R. (1986) Spatial orientation by familiarity cues. *Proceedings of the 6th European Annual Conference on Manual Control,* June, 1986, University of Keele, Great Britain.

Grunwald, Arthur, & Ellis, Stephen R. (1988) Interactive orbital proximity operations planning system. *NASA TP 2839* , November, 1988.

Helmholtz, H. *Handbook of physiological optics* (1856-1866) Southall trans. Optical Society of America (1924), Rochester, N.Y..

Herbst, P.J., Wolff, D.E., Ewing, D. and Jones, L.R. (1946, February). The TELERAN proposal. *Electronics, 19,* 125-127.

Howard, Ian. (1982) *Human visual orientation.* Wiley, New York.

Jenks, George F. and Brown, D. A. (1966). Three dimensional map construction. *Science,* 154, 857-846.

Kim, W. S., Ellis,S.R., Tyler, M., Hannaford,B., & Stark, L. (1987) A quantitative evaluation of perspective and stereoscopic displays in three-axis manual tracking tasks, *IEEE Trans. on Systems Man and Cybernetics, SMC- 17,*61-71.8

McGreevy, Michael W., Ellis, Stephen R. (1986) The effects of perspective geometry on judged direction in spatial information instruments. *Human Factors, 28,* pp. 421-438.

Meagher, Donald J. (1987) The manipulation, analysis and display of 3D medical objects using octree encoding techniques. *Innovation et Technology en Biologie et Medecine,* 8, Special 1, 21-36.

Muybridge, Eduward. (1975) *Animals in Motion,* Lewis S Brown, ed., Dover, New York.

Nagata, S. (1986) How to reinforce perception of depth in single two-dimensional pictures. Selected papers in basic researches, No 44-51, *Proceedings of the 1984 SID*, 25, No. 3, 239-246.

Parker, D.E., Renschke, M.F., Arrott, A.P., Homick, J., Lichtenberg, B. (1986) Otolyth tilt-translation reinterpretation following prolonger weightlessness: implications for preflight training. *Aviation, Space, and Environmental Medicine, 56,* 601-606.

Piaget, J, and Inhelder, B (1956) *The childs conception of space.* Routledge and Kegan Paul, London.

Roscoe, S.N. (1984) Judgements of size and distance with imaging displays. *Human Factors, 26,* 617-629..

Roscoe, S.N. (1987) The trouble with HUDs and HMDs. *Human Factors Society Bulletin, 30, 7,* 1-3.

Russell, Gregory, Miles, Richard B. (1987) Display and perception of 3-D space-filling data. *Applied Optics,* 26, 973-982.

Sutherland, Ivan E. (1970) Computer displays. *Scientific American,* 222, 57-81.

Velger, Mordekai. (1988) Adaptive filtering of biodynamic stick feed-through in manipulation tasks on board moving platforms. *Journal of Guidance and Control Dynamics, 11,* 153-158

Weintraub, D.J., Haines, R.F., Randle, R.J. (1985) Head up display: HUD utility II: runway to HUD. *Proceedings of the 29th Meeting of the Human Factors Society.* University of Michigan, Ann Arbor.

Wells, Maxwell, Griffin, Michael J. (January, 1984) Benefits of helmet-mounted display image stabilisation under whole body vibration. *Aviation, Space and Environmental Medicine,* 13-18

USE OF GRAPHICS IN DECISION AIDS FOR TELEROBOTIC CONTROL
(Parts 5-8 of an 8-Part MIT Progress Report)

Thomas B. Sheridan, James B. Roseborough, Hari Das, Kan-Ping Chin, and Seiichi Inoue

Man-Machine Systems Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

This paper summarizes four separate projects recently completed or in progress at the MIT Man-Machine Systems Laboratory. Four others are described in a companion paper in Volume I.

5. A DECISION AID FOR RETRIEVING A TUMBLING SATELLITE IN SPACE -
James B. Roseborough

Abstract. A decision aid for retrieving a tumbling satellite in space was constructed and tested in the laboratory. It was found that, though a perfect aid improves decision making, an aid whose modeling errors are nearly equal to those of the human operator will degrade performance. In addition, an aid presenting only point estimates is superior to an aid that also presents uncertainties.

Introduction. In the future, servicing of satellites in space may become more cost effective than simply replacing malfunctioning satellites with new ones. Retrieving a satellite for servicing can be a problem, however, especially if the satellite is nutating or tumbling. A decision aiding system was constructed for this situation that will help astronauts in this and other retrieval tasks.

The present approach to this problem is described in Rice et.al. [1] and Hartley et. al. [2]. In it, a vehicle such as a Manned Maneuverable Unit (MMU) is first positioned near the target satellite. Then, the MMU circles the target at the same rotation rate and about the same axis as the target. Finally the MMU closes in on the target until the grappling fixture of the target is secured to the MMU. Once this is done the MMU can be used to slow the rotation of the target with its thrusters. Experiments have shown that the task performed in this way is difficult, especially for complex motion like nutation and tumbling, and the Solar Max mission confirmed the problems in this approach.

Decision aid design. A decision aid has been designed [3] in which normatively derived state estimates are presented to the human operator. Figure 1 shows the elements of this system.

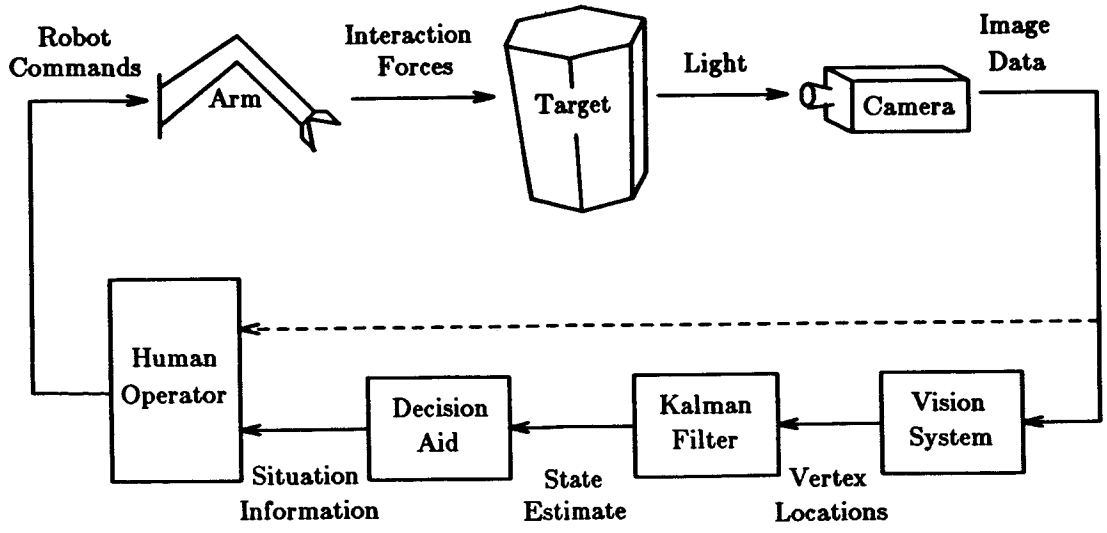


Figure 1. The components of a satellite retrieval decision aid based on state estimates.

In our conception of the retrieval mission, the satellite first achieves a parking orbit adjacent to the target satellite. Then a decision support system is used to build up a model of the rotational motion of the satellite in a computer. The operator can work with this model, using it to predict future states of the target or look at past states. Finally, the operator will use a robotic arm to reach out and grapple the satellite as it rotates, using his decision aid to help him select a good opportunity for mission success. Alternatively, the operator gives only the final signal to an automatic arm control and retrieval subsystem.

The information provided by the decision aid was cosidered along three dimensions: 1) what variables were displayed, 2) over what timescales they were shown, and 3) what statistical information was included. Displays included both pictographic images, in which a measured position was displayed directly, and derived decision variables. For example, we defined an objective function that can be loosely called "grappleability," or simply "goodness." This function conveys in a single variable the degree to which a certain process state represents an ideal grappling opportunity. This function reduces the information demands on the operator by transforming the decision problem from that of observing and processing six independent variables to that of processing a single process state indication, grappleability. Also, an acceptability display, was available which presented the probability that the orientation would be within a certain range of orientations that are acceptable for grappling.

For this problem, past as well as future values were provided. Past values give the operator a sense of the overall statistics of the process, while future predicitions provide the operator with uncertain estimates of where the process will be in over a prediciton interval. Predictor displays having three different resolutions were used in the experiment. These were called historic, long term, short term, and `8-12` second displays.

To behave normatively, the operator must hold a *belief state,* which is a distribution over the state space of the process. While this is impossible for most reasonable systems, a first order approximation such as that provided by a Kalman filter can be used by the computer and communicated to the operator. Our decision aid provided both information with varying levels of statistical detail.

Experiments and Results. Experiments were performed [3] using the decision aid in various configurations to determine which types of information were most useful. Regardless of the specific performance measure used, the perfect aid was superior to no aid, and degraded aids were worse than no aid. It is noteworthy that in both local and global terms, point estimates produced better decision making than displays that used uncertainties. A possible explanation is that when presented with statistical information, the human will first derive point estimates, so by having the computer provide only point estimates, workload is reduced and accuracy is increased.

Another interesting result is the relation of global performance (error) to motion complexity, or degrees of freedom, as in Figure 2. Here it seen that for simple tasks, that is, one DOF tasks, the decision aid is not necessary, and even reduces decision performance slightly. For difficult cases of two and three DOF motion, the decision aid was useful, since the human has no ready algorithms for computing the long term future states of the process.

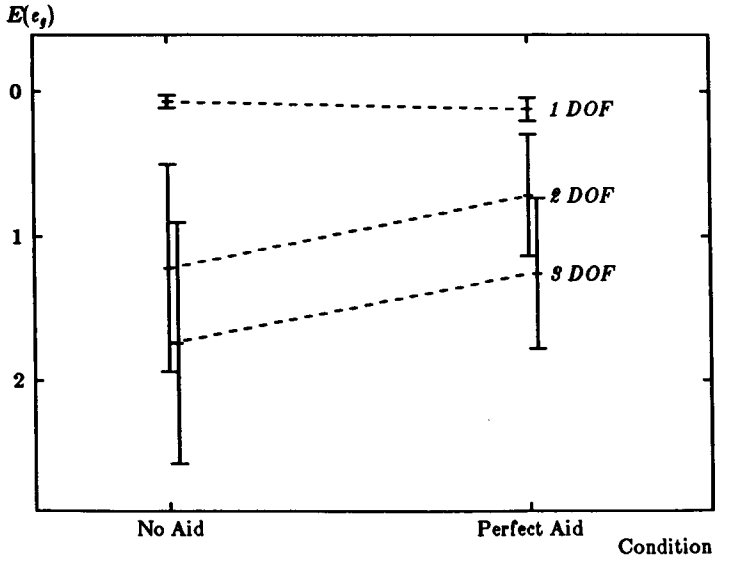


Figure 2. Global squared error as a function of the decision aid and the motion drgrees of freedom.

An interesting comment was made by one subject regarding the use of low quality information provided by the decision aid. When asked if he thought it helped him, he said,
"My guess is it did, but I am not sure. Whenever the information was there I just wanted to use it."

This suggests a picture of the displayed information as being an active element, rather than the having the passive, inert nature that it is most often given. Further support is found when we recognize that the presence of predictive displays resulted in hasty decisions.

The following qualitative observations were also made:
(1) The task is actively reshaped in terms of the available measurements. If the probability of an acceptable range of endpoint angles is displayed, the subject will perform the task in a way that tends to maximize this probability.
(2) Much of the human's activity in the task was directed towards finding patterns in the environment. Frequently historic displays were used to examine presence or absence of periodicity - a concept that is not explicitly modeled by the state-estimation-based decision aid.
(3) Information about past values is less active than information about future values. Subjects were less likely to make hasty decisions when using historic information than when using predictive information.
(4) The decision aid can have significant emotional effects on the user. For example, one subject reported that the aid gave him a feeling of confidence in making his decisions, and allowed him to manage his own attention resources much better than when it was not available. Another example is the undue sense of urgency that caused hasty actions in several cases. As these ultimately affect the quality of the decisions made, they should not be passed over in real systems.

References

(1) Hartley, C.S., Cwynar, D.J. Garcia, K.D. and Schein, R.E., Capture of satellites having rotational motion, *Proc. Human Factors Soc. 30th Ann. Meeting*, 1986.
(2) Rice, J.R., Torchak, J.P., and Hartley, C.S., Planning for unanticipated satellite servicing operations, *Proc. Human Factors Soc. 30th Ann. Meeting*, 1986.
(3) Roseborough, J.B., *Aiding Human Operators with State Estimates*, PhD Thesis, MIT, Cambridge, MA, May 1988.

6. KINEMATIC CONTROL AND GRAPHIC DISPLAY OF REDUNDANT TELEOPERATORS - Hari Das

Abstract. The goal of this work is to help the operator of a redundant teleoperator perform end effector positioning and orienting tasks in a three dimensional world while avoiding collision between the teleoperator and obstacles in the environment.

Introduction. We limit ourselves to:
1) situations in which the operator does not have direct view of the teleoperator and has to use a displayed view to perform tasks,
2) solving the kinematic problem (we do not address the dynamics and control problem) of attaining the desired end effector position and orientation while keeping all parts of the teleoperator away from obstacles,
3) improving the operator's perception of the environment and the location of the teleoperator in it with an improved visual display, and
4) showing, with simulation experiments on human subjects, that our proposals achieve the goal .

We develop an interface between the operator and the teleoperator that is designed to give the operator good visual sense of the environment and to enable simple instruction by the operator on desired trajectories for the teleoperator.

The unique features of this work are the numerical inverse kinematic algorithm for handling kinematically redundant teleoperators to reduce the workload on the operator, and a technique for modelling the environment to enable its display from various viewpoints. Our experiments on human operator performance show that the display greatly affects the ability to perform tasks well. The idea of automatically determining the best view to display is also explored.

Kinematic Control. The method chosen in our work is a numerical inverse kinematic approach. The operator specifies end effector paths while the computer determines the best configuration for the teleoperator that maintains the end effector at the specified position and orientation while keeping other parts of the teleoperator as distant from obstacles as possible. Path planning is a cooperative effort between the operator and the computer aid.

The numerical method we use to solve the inverse kinematics is to first formulate the problem as a constrained optimization, then use a hybrid of the generalized inverse and the method of steepest descent to solve for joint positions (Das, 1989).

Visual Display. We propose the creation, from sensor information, of a computer data structure to represent objects in the environment. In this scenario, as more sensor data is received, the data structure is updated to improve the model. A view of the world chosen by the operator and represented in the computer can then be displayed on a graphic screen (see Fig. 3). Advantages of representing the environment in such a data structure are:
1) views of the model can be drawn as seen from any point in space.
2) any geometric information on objects in the environment may be processed from the data.
3) the reconstructed environment may be displayed visually to the operator while a camera image in poor lighting or visibility conditions may not provide much information.

We assume that a model of the environment is available. The problem of processing sensor information to form a model of objects in the environment has been pursued by others (Winey, 1981, Marce and Even, 1988). We have identified some elements of a good view and have developed an algorithm to select it. The determination of a good view is based on the idea that it is desirable to have closest distances between the teleoperator and obstacles orthogonal to the line of sight (Das, 1989).

Experiments. A simulation of a twelve d.o.f. vehicle-manipulator teleoperator (6 d.o.f. wrist-partitioned manipulator on a 6 d.o.f. vehicle) has been developed to test the proposals suggested in this work. In experiments, the performance of human subjects on positioning and orienting the end effector at specified locations in a 3 dimensional space among a field of obstacles was measured. Time taken to complete the tasks and number of collisions with obstacles while performing tasks were used to determine performance.

Experiments have been conducted, comparing the inverse kinematic control with the operator having a 6 d.o.f. end effector position and orientation input to a fully manual control method with the operator having a 12 d.o.f. input. Two six d.o.f. input devices were used, one to position and orient the vehicle and the other to position and orient the end effector with respect to the vehicle. A closed-form solution was used to solve the inverse kinematics of the wrist-partitioned manipulator.

In addition, human subject performance with an automatically selected view was compared to an operator selected view, a simulated view from a fixed point in the 3 dimensional space and a simulated view from the vehicle.

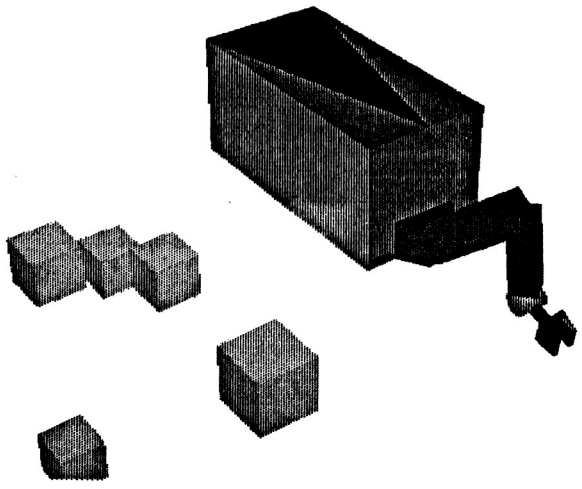


Figure 3. Vehicle-manipulator system as seen by human operator.

Conclusions. Briefly, the results from our experiments indicate:
1) Performance is better when the human subjects used the 6 d.o.f. end effector positioning and orienting method.
2) The different options on viewing greatly affected human subjects' performance.
3) Performance was best when the subjects were able to select a view.
4) Performance with the automatically selected view was better than with the simulated views either from the vehicle or from a fixed point.

References
(1) Das, H., *Kinematic Control and Visual Display of Redundant Teleoperators,*
Ph. D. thesis in progress, MIT., Jan., 1989.
(2) Winey, C. M., *Computer Simulated Visual and Tactile Feedback as an Aid to Manipulator and Vehicle Control,*
M.S. Thesis, MIT, June, 1981.
(3) Even, P. and Marce, L., Manned Geometric Modelling for Computer Aided Teleoperation, *Proc. Int. Symp. on Teleoperation and Control,* Bristol, England, July, 1988.

7. REAL-TIME TERRAIN / OBJECT GENERATION: A QUAD-TREE APPROACH -

Kan-Ping Chin

Abstract. A hidden surface removal algorithm for a projected grid surface is developed to accelerate the display of 3-dimensional terrain or object. The size of the database is reduced to about one fifth of the database with constant sampling resolution; and the frame speed of the system is about four times faster.

Introduction. A realistic 3-dimensional computer-graphic presentation is important for training the operator of a teleoperated vehicle [1] or for control based on a computer model. The feeling of realism depends on two factors : (1) a realistic image, and (2) a real-time display of the image. A realistic terrain image usually requires a very large database to record appropriate geometric terrain height information at latitude and longitude in a regular reference grid (coordinates of each sample point); however, the manipulation of a large database is very time consuming. On the other hand, a small database which samples at sparser points of a terrain can be displayed very fast, but may sacrifice the realism of the image. Thus, there is a conflict in terms of computer graphics between these two aspects: the realistic instantaneous image, and the real-time display of the image.

A method has been developed to reduce the database size without sacrificing realism in the image. In this method, instead of sampling height at regularly spread points of the horizontal grid of the terrain, only the points where the gradient changes most rapidly are sampled. In this way, the database size is reduced. A quad-tree data structure is used to store the sampled data. A recursive algorithm is developed to retrieve the data from the quad-tree and to display it in a specific order so that the hidden surfaces are also removed.

Sampling Data at Different Resolutions. As Attneave [2] suggested, the points of concentrated information in figures are where their gradients change most rapidly. Therefore, common objects may be represented with great economy and fair fidelity by marking the points around the sharpest gradient change (i.e. small radius of curvature) and then connecting these points with straight lines. As shown in Figure 4, the recognizability of the object indicates that most of the important information has been retained.

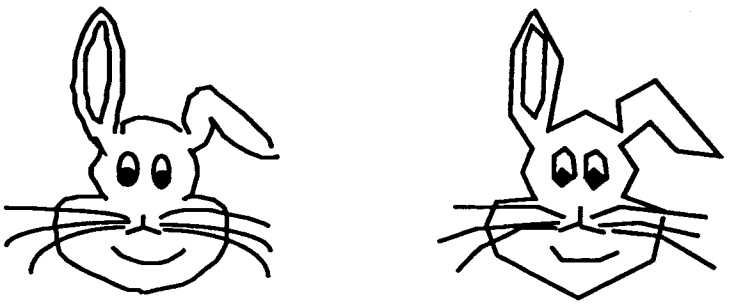


Figure 4. A curvilinear object represented by straight lines.

Applying the above concept, a mountain range that has varying gradient is intensively sampled; and a plain, being regarded as featureless, is sparsely sampled. In this way, a sampled database is smaller than the one with constant sampling distance. For a natural scenery such as terrain, this averaging over "texture" of the figure is analogous to what happens when a halftone photograph is recopied several times on high contrast paper. However, since the data is irregularly scattered if we sample it at various resolutions, we need a special data structure to handle it.

Quad-tree Data Structure. A quad-tree data structure[3], often used in image processing, can hierarchically store graphic data of a picture at various resolutions. A square picture can be divided into four sub-squares, or not divided at all, according to a predefined criterion such as, in this case, curvature of the picture; the sub-squares can be iteratively further subdivided if necessary. In this way, variable resolution of a picture are efficiently stored in a hierarchical, tree-like structure.

The place to store the graphic information of the picture is called a "node". Each node either has four branches which correspond to the four subdivided pictures, or has no branch at all; accordingly, either the addresses of the child branches, or the information of the picture if there is no child branch, is stored at each node (Fig. 5).

To retrieve the pictorial information, we have to check the availability of the information at each node first. If the information of a picture is stored at a node, then we can retrieve it and generate the picture from the data retrieved. If there is no pictorial information in the node, we then use the addresses of the child branches to check the availability of data at each child branch repetitively. The procedure described is applicable to nodes everywhere in the quad tree; therefore, a subroutine calling itself recursively is used to walk through the tree.

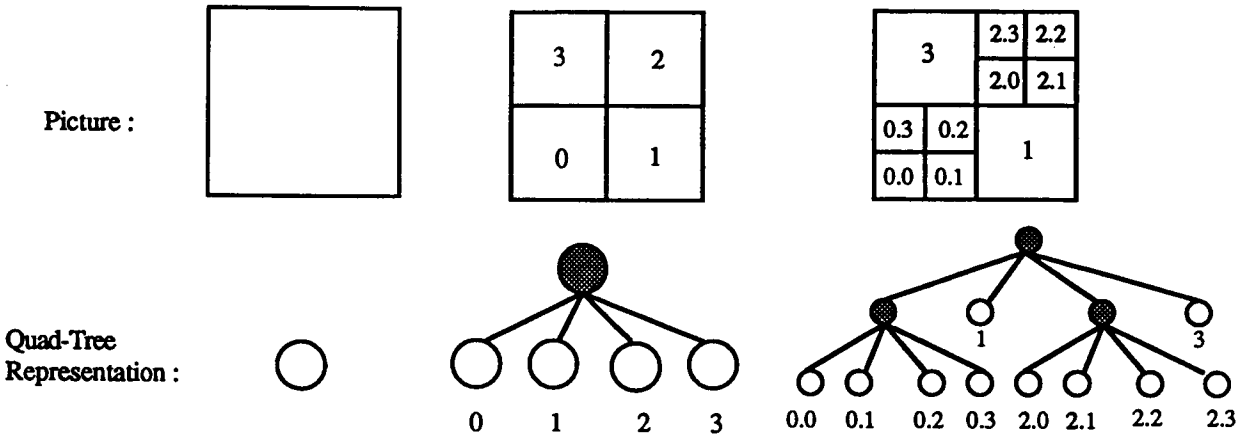


Figure 5. A quad-tree representation of a 4 X 4 "picture".

Hidden Surface Removal Algorithm for Projected Grid Surface. For a bivariate (in the form of z=f(x,y)) grid surface, which we used to model the terrain, the projection of any facet is bounded by the projection of the boundary of the facet. This geometric property provides a simple way to enumerate the facets from far to near for a given view point [4]. Consequently, two of the most time consuming jobs in hidden surface removal algorithm, to compute the distance of objects to the viewing position and to sort those distances to determine the display sequence, are not essential anymore. That is why the present method, without doing these two jobs, is very fast compared with other methods.

We distinguish three different types of viewer location: the viewer sitting on top of the terrain, the viewer looking at the terrain from an edge, and the viewer looking at the terrain from a corner. As shown in Figure 6, for case 1, the viewer on top of the terrain, the terrain can be separated into nine regions and the viewer will be looking from region 0. The algorithm can be divided into 3 steps. Each step processes a specific group of regions. In each region, the facets are always processed row by row for display from the farthest one from the viewer to the nearest one, as depicted by the arrows in the picture. The other two cases, where the viewer is looking from an edge or from a corner of the terrain, are only special instances of case 1, and therefore can be solved by part of the above procedure. For example, if the viewer is looking from the lower edge of the terrain, the terrain can be separated into three regions and their display sequences are the same as in case 1.

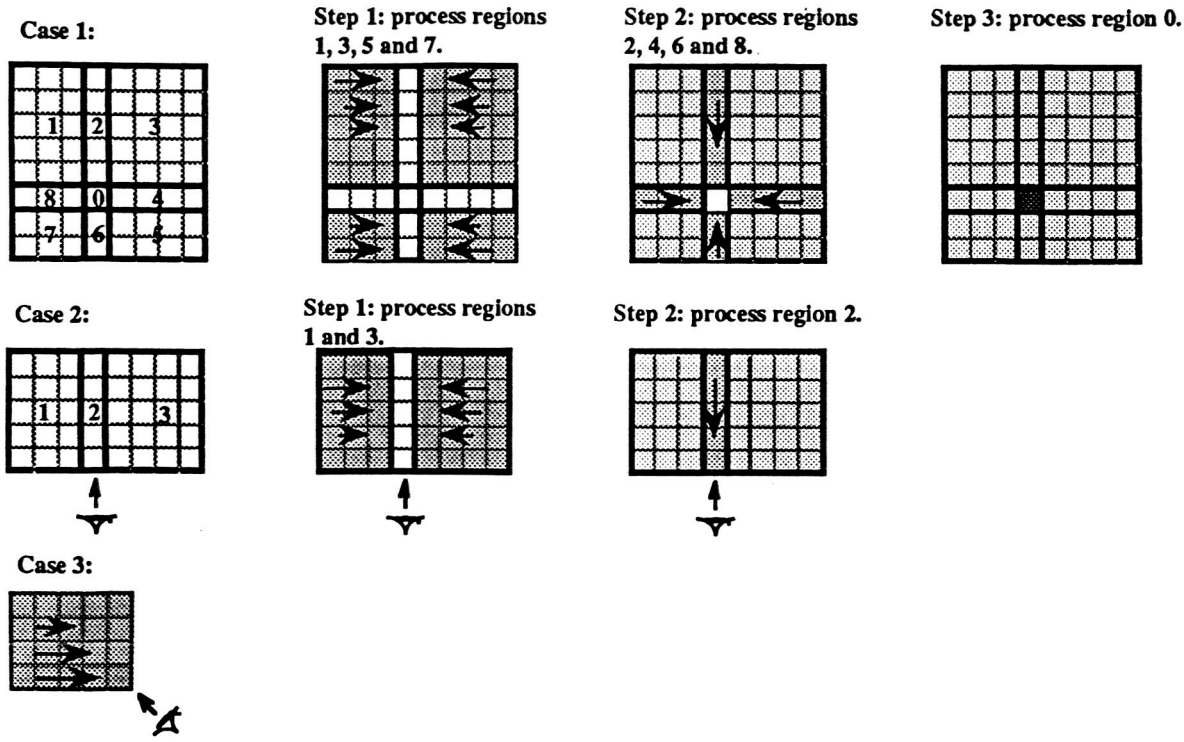


Figure 6. Hidden surface removal for a gridded surface. Case 1: viewer sitting on top of region 9. Case 2: viewer looking from bottom edge. Case 3: viewer looking from a corner.

Applying this method to the quad-tree data structure, each facet is represented by a tree. Each sub-facet (branch of the tree) must also be processed from far to near for each region. We can produce a table for the display sequence of the sub-facets for each region.

Conclusion. This method has been implemented in a C language computer program on an IRIS 2400 graphic workstation. The database is sampled manually from a real-world contour map. By sampling at different resolutions for different areas, the database is reduced to about one-fifth (from 2,048 patches to 416 patches) of the one with constant sampling resolution (the ratio may vary depending on the shape of the terrain.); and the frame speed of the system is about four times faster (from 3 frames per second to 13 frames per second). Gouroud (smooth) shading can be implemented to enhance the realism of the image; however, it decreases the frame speed to about 3 frames per second.

The terrain is integrated with a simulator which features the dynamics of a telerobot and associated structure. The simulator is about 9 frames per second without smooth shading and about 2.4 frames per second with Gouroud smooth shading.[5].

References

[1] Sheridan, T.B., *Improving the Effectiveness of Remotely Operated Vehicles for Inspection, Maintenance and Repair of Offshore Structures through Simulation Aids.* MIT Sea Grant Program, 1985.

[2] Attneave, F. Some informational aspects of visual perception, *Psychological Review*, Vol. 61, No. 3, 1954.

[3] Hunter, G.M. and Steiglitz, K. Operations on images using quadtrees, *IEEE Trans. on Pattern Analysis and Machine Intell.*, V1, No. 2 pp. 145-153, April 1979.

[4] Anderson, D.P. Hidden Line Elimination in Projected Grid Surfaces, *ACM Transactions on Graphics*, Vol. 1, No. 4, pp. 274-288, October 1982.

[5] Chin, K.-P., *Human-Interactive Simulator for Underwater Remotely Operated Vehicles*, SM Thesis, Dept. of Mechanical Engineering, MIT, 1988.

8. TWO DIMENSIONAL CONTROL FOR THREE DIMENSIONAL OBSTACLE AVOIDANCE - Seiichi Inoue

Abstract. A technique has been developed by which a two-dimensional display can be used for guiding a hand or vehicle and ensuring that there is no collision in three-space. The human operator, using only a mouse (and cursor on his 2-D computer display), locates key points of 3-D obstacles, as seen by two cameras. The computer then indicates the intersection boundaries for any selected plane from the start point to the goal or subgoal. The operator can then select a robot hand trajectory on the plane by using the mouse and cursor and thereby guarantee obstacle avoidance.

Key Point Measurement of 3-D Obstacle. First we have to know the position of arbitrary points in 3-D space. Two TV cameras (mounted parallel to each other on the vehicle) are used to measure the projections of the object on the 2-D image plane.

Let 3-D points in task environment and 2-D points on either TV image plane (screen) be represented respectively by [x, y, z] and [x*, y*]. The relation between these points can be written as follows, where H is a scale factor.

$$[x \;\; y \;\; z \;\; 1]\, T = H\, [x^* \;\; y^* \;\; 0 \;\; 1] \quad \text{.....(1)}$$

The perspective transformation matrix T is

$$T = \begin{vmatrix} T_{11} & T_{12} & 0 & T_{14} \\ T_{21} & T_{22} & 0 & T_{24} \\ T_{31} & T_{32} & 0 & T_{34} \\ T_{41} & T_{42} & 0 & T_{44} \end{vmatrix}$$

Eliminating H yields

$$(T_{11} - T_{14}\, x^*)\, x + (T_{21} - T_{24}\, x^*)\, y + (T_{31} - T_{34}\, x^*)\, z + (T_{41} - T_{44}\, x^*) = 0 \quad \text{.....(2)}$$

$$(T_{12} - T_{14}\, y^*)\, x + (T_{22} - T_{24}\, y^*)\, y + (T_{32} - T_{34}\, y^*)\, z + (T_{42} - T_{44}\, y^*) = 0 \quad \text{.....(3)}$$

When two cameras are used, and the perspective transformation matrices are represented respectively by T1 and T2, the following simultaneous equations are obtained from equations (2) and (3).

$$(T_{111} - T_{141}\, x^{*1})\, x + (T_{211} - T_{241}\, x^{*1})\, y + (T_{311} - T_{341}\, x^{*1})\, z + (T_{411} - T_{441}\, x^{*1}) = 0 \quad \text{.....(4)}$$

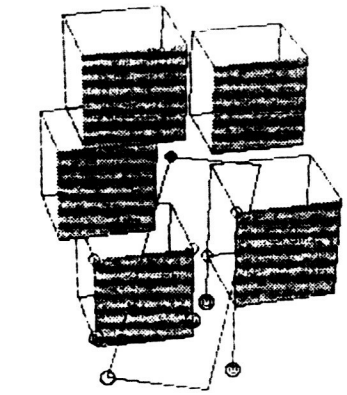
$$(T_{121} - T_{141}\, y^{*1})\, x + (T_{221} - T_{241}\, y^{*1})\, y + (T_{321} - T_{341}\, y^{*1})\, z + (T_{421} - T_{441}\, y^{*1}) = 0 \quad \text{.....(5)}$$

$$(T_{112} - T_{142}\, x^{*2})\, x + (T_{212} - T_{242}\, x^{*2})\, y + (T_{312} - T_{342}\, x^{*2})\, z + (T_{412} - T_{442}\, x^{*2}) = 0 \quad \text{.....(6)}$$

If matrices T1 and T2 and x*1, x*2, y*1 are known, the 3-D position x, y, z is obtained by use of equations (4), (5), and (6).

Avoidance of Obstacles. By use of the preceding method we can know the 3D positions of arbitrary points on a video display. Accordingly the operator, using a mouse and a cursor, locates key points [x*1, x*2, y*1] of 3D obstacles on the 2D display. The computer then indicates the intersection point of an obstacle and any selected plane which includes the start point and the goal or subgoal.

Figure 7 shows, using an example of a task environment and the plane which an end-effector moves on, that obstacle A and plane C intersect, and obstacle B is above plane C. Anywhere the end effector moves on plane C, the operator can easily avoid the obstacle A.



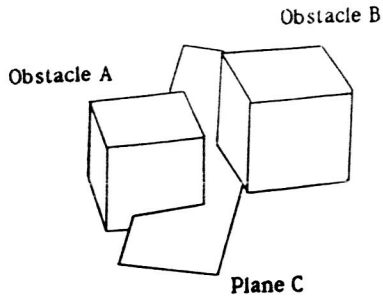
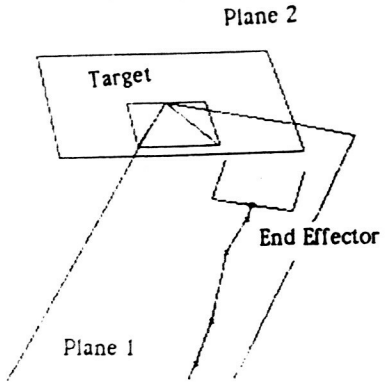


Figure 7. Task environment and intersection points of plane and obstacle.



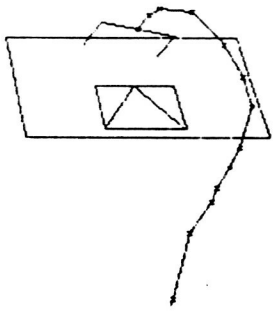


Figure 8. Moving end effector from plane 1 to plane 2.

Controlling the End Effector on the Plane. When the end effector moves on the plane from start point to the goal, we can have position control and one-axis rotation control in 3-D space by use of a 2-D input device like the mouse. If the desired position (x*, y*) for the end effector on the display is decided upon, then the 3-D position (x, y, z) of it can be calculated.

Not only robot hand or other end effector translation in 3-D space, but also rotation of it about the perpendicular axis to the plane, can be controlled by use of a 2-D input device. For example, one must control the rotation of the end effector when one is ready to grasp the target. Then another plane included the target can be set up, and the end effector can be transferred to the new plane, as shown Figure 8. After transfer to another plane, it is very easy for the end effector to approach and grasp the target.

Reference.

Inoue, S. and Hashimoto, T., 3-D Data Measurement with a Single Camera in an Arbitrary Position, *Research Reports of Fukui College of Technology*, Japan, 1986

ACKNOWLEDEGMENT. Projects 1 and 2 were sponsored by Jet Propulsion Laboratory, Project 3 by Woods Hole Oceanographic Institution, and Project 4 by the Japanese Government. Their support is gratefully acknowledged.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 89-7	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle PROCEEDINGS OF THE NASA CONFERENCE ON SPACE TELEROBOTICS (VOLUMES I-V)		5. Report Date January 31, 1989 6. Performing Organization Code
7. Author(s) G. Rodriguez and H. Seraji (editors)		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

These proceedings contain papers presented at the NASA Conference on Space Telerobotics held in Pasadena, January 31 - February 2, 1989. The Conference was sponsored by the NASA Office of Aeronautics and Space Technology, together with ARC, LRC, GSFC, JSC, MSFC, KSC and JPL. The theme of the Conference was man-machine collaboration in space. The Conference provided a forum for researchers and engineers to exchange ideas on the research and development required for application of telerobotics technology to the space systems planned for the 1990s and beyond. The Conference: (i) provided a view of current NASA telerobotic research and development; (ii) stimulated technical exchange on man-machine systems, manipulator control, machine sensing, machine intelligence, concurrent computation, and system architectures; and (iii) identified important unsolved problems of current interest which can be dealt with by future research. There were about 500 international participants including about 100 from abroad.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Engineering	Unclassified; unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	2,386	